Regular Manuscript

Accelerating Singular Spectrum Transformation for Scalable Change Point Detection

Submission ID c797ca46-9182-4103-911e-f3c5e264f201

Submission Version Initial Submission

PDF Generation 31 Oct 2025 17:36:58 EST by Atypon ReX

Authors

Mr. LUCAS WEBER
Corresponding Author
Submitting Author

Affiliations

- Department of Computer Science, Friedrich-Alexander Universität Erlangen-Nürnberg

Prof. RICHARD LENZ

Affiliations

- Department of Computer Science, Friedrich-Alexander Universität Erlangen-Nürnberg

Additional Information

Subject Category

Computers and information processing

Industry applications

Science - general

Signal processing

Keywords

Algorithmic efficiency

Computational efficiency

Error analysis

Linear algebra

Signal analysis

Signal processing algorithms

Singular value decomposition

Select a Manuscript Type

Research Article

Files for peer review

All files submitted by the author for peer review are listed below. Files that could not be converted to PDF are indicated; reviewers are able to access them online.

Name	Type of File	Size	Page
access.pdf	Main Document - PDF	770.2 KB	Page 4
access.pdf	Author's Tracked Changes (Highlighted Manuscript of Changes)	1.3 MB	Page 25
IEEE-Access-Response-to-Reviewers.pdf	Author Response	261.4 KB	Page 46

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Accelerating Singular Spectrum Transformation for Scalable Change Point Detection

LUCAS WEBER[1], **RICHARD LENZ**[1]
[1]Department of Computer Science, Friedrich-Alexander Universität Erlangen-Nürnberg, 91058 Erlangen, Germany
Corresponding author: Lucas Weber (e-mail: lucas.weber@fau.de).

SIEMENS Energy AG has supported this work in the context of Project DARTS.

ABSTRACT The Singular Spectrum Transformation (SST) is a powerful technique for Change Point Detection (CPD) and is widely applied in time series analysis to identify abrupt structural changes. Due to the involvement of the Singular Value Decomposition (SVD), the runtime of the SST cubically grows ($\mathcal{O}(N^3)$) with the window size N. Therefore, applications building on the SST are often limited to small window sizes. An improved version of the SST, the IKA-SST, removes this bottleneck, but still suffers from similar scaling problems. Utilizing randomized decompositions and a fast matrix product, we reduce the computational complexities of both the SST and the IKA-SST to log-linear ($\mathcal{O}(N \log N)$), enabling efficient CPD for large window sizes and high-frequency signals.
Focusing on the approximation error, we discuss how randomized decompositions are well-suited to SST-based CPD. By linking the scoring error to the decomposition error via a novel error bound, we provide theoretically grounded arguments that the efficiency gains incur only a small approximation error. We empirically evaluate the improved SST algorithms on a diverse collection of real and synthetic time series. Comparisons with the original algorithms confirm that the speedup, parallelization, and low approximation error translate well from theory to practice. Depending on the application, we measure acceleration factors of up to three orders of magnitude, reducing the wall time from hours to minutes, or even seconds, broadening the applicability of SST-based CPD. To support reproducibility and further applications, we publish our experimental code and reference implementations.

INDEX TERMS Change Point Detection (CPD), Signal Processing, Singular Spectrum Analysis (SSA), Hankel Matrix Decomposition, Randomized Singular Value Decomposition (rSVD).

I. INTRODUCTION

Time series analysis is paramount in data science and analytics because it reveals patterns and trends within chronological data and drives informed decision-making processes across various industries. Change Point Detection (CPD), as a research direction in time series analysis, provides methods to find abrupt structural changes in time series, which can be used to detect state changes in the measured systems. CPD finds application in a wide range of real-world problems such as medical condition monitoring, climate change analysis, speech recognition, and human activity analysis [1].
The Singular Spectrum Transformation (SST) [2], [3] is a common CPD algorithm that detects changes in time series shape or patterns. Example applications of the SST include power plants with thousands of time series [4] and high-frequency measurements from electrical machines [5]. The SST estimates a change score by comparing the similarity of characteristic time series components. It derives these components by performing the Singular Value Decomposition (SVD) on a Hankel matrix, constructed from the original time series, and parametrized by a window size (see fig. 1). The window size N is the main hyperparameter of the SST and directly determines the matrix size for the SVD. The computational complexity and, therefore, the runtime of the SVD scales cubically with the matrix size ($\mathcal{O}(N^3)$), increasing rapidly for larger window sizes. The IKA-SST [3] is an improved variant of the SST that removes SVD as the computational bottleneck but requires the computation of a correlation matrix. Similar to the SVD, this step scales poorly with the window size. See Section III-C for further details on both algorithms. Addressing these bottlenecks is the focus of our contribution.

The Problem. In the case of larger patterns or high-frequency signals, the window size increases and the SST runtime quickly becomes infeasible. In practice, this substantially

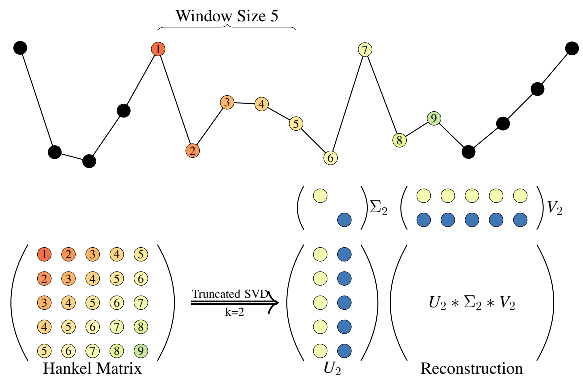


FIGURE 1: Visualization of Hankel matrix construction from a random time series (window size = 5) and truncated SVD reconstruction ($k = 2$).

raises the resource requirements, entails elaborate preprocessing steps, or limits the practical applicability of the SST to smaller patterns or fewer signals.

The Goal. Speed up SST-based change point scoring for increasing window sizes. Specifically, we aim to reduce the runtime of both the SST and IKA-SST to speed up existing applications and broaden their applicability to high-frequency signals and long-duration patterns. We seek to maintain algorithmic structure while reducing computational effort.

To achieve this goal, we employ approximative methods from linear algebra and exploit the special Hankel structure of matrices occurring in both the SST and IKA-SST. In particular, we propose the application of the randomized Singular Value Decomposition (rSVD), a fairly recent advancement from the field of randomized linear algebra, to speed up and parallelize the SVD as a core component of the SST. Combining these methods with a fast Hankel matrix product reduces the computational complexity of both methods from $\mathcal{O}(N^3)$ to $\mathcal{O}(N \log N)$. We demonstrate how these asymptotic complexities translate to runtime improvements for growing window sizes and provide theoretical arguments for our algorithmic choices. We substantiate these arguments with empirical measurements on an extensive dataset of real and synthetic time series. The main contributions of this paper can be summarized as follows:

- **SST and IKA-SST with log-linear time- and linear memory complexity.** We reduce the per-step computational complexity of the SST [2] and IKA-SST [3] from $\mathcal{O}(N^3)$ to $\mathcal{O}(q(k+p)N \log N)$, and reduce the space complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}((k+p)N)$. Over a length-L time series, this yields $\mathcal{O}(Lq(k+p)N \log N)$ (SST) and $\mathcal{O}(LkN \log N)$ (IKA-SST) with parameters $k, p, q \ll N$. This change in complexity results in faster computation times, and extends the applicability of both SST-variants to larger window sizes and high-frequency signals.
- **Suitability and Expected Error Bound.** By discussing the spectral properties of Hankel matrices and linking the scoring error with the decomposition error via a novel error bound, we provide theoretically grounded arguments for the suitability of the rSVD for general Hankel matrices, and SST-based CPD specifically.
- **Empirical Evaluation.** On extensive real and synthetic data, we confirm our theoretical claims, demonstrating low rSVD approximation errors and low end-to-end scoring errors. We demonstrate parallel scaling and strong runtime improvements that grow with N. For example, our improved IKA-SST is 814x faster at $N = 5000$. In a small case-study, our algorithmic modifications reduce the runtime from 22 hours to 22 minutes.
- **Artifacts.** We release reference implementations of the (improved) SST algorithms and our measurement code.

The remainder of this paper is structured as follows: We first introduce the related work. We explain our selection criteria for the related work (Sec. II), summarize their contributions and then distinguish our work from it. Following this, we discuss our methods (Sec. III), establishing all necessary concepts first (Sec. III-A & III-B). We explain both SST variants (Sec. III-C), and the rSVD (Sec. III-D) algorithm in detail and introduce a fast matrix product for Hankel matrices (Sec. III-E). We then show how the rSVD and the fast matrix product can be used to improve the SST and the IKA-SST respectively (Sec. III-F & III-G). In the last part of the methods section, we provide arguments for the suitability of the rSVD in terms of the approximation error (Sec. III-H), relying on spectral properties of the Hankel matrix and derive an error bound connecting the decomposition accuracy with the scoring error (Sec. III-I). In the evaluation section (Sec. IV), we introduce our dataset (Sec. IV-A) and provide implementation details (Sec. IV-B). Using the dataset, we empirically substantiate our assumptions on the spectral properties of Hankel matrices, measuring the singular value decay of Hankel matrices extracted from diverse time series (Sec. IV-C). We then empirically evaluate the approximation error of the rSVD (Sec. IV-D) and how it affects the final change score when built into the SST (Sec. IV-E). In this context, we also show how we derive several hyperparameters for the rSVD by employing other research on time series structure. In the same section, we empirically demonstrate that the fast Hankel matrix product does not introduce large numerical errors. As the last part of the evaluation section, we address our overall goal by evaluating how the reduced computational complexity translates to actual runtime improvements for all algorithms (Sec. IV-F). In the discussion (Sec. V), we reflect on the empirical results, discuss their implications, and the limitations of our improvement. Finally, we give an outlook on potential further improvements (Sec. V-B) and conclude the paper (Sec. VI).

To make our research reproducible and accessible, the source code for this paper and the datasets are available online[1].

[1]https://github.com/Lucew/approximate_hankel

While the repository contains the research code dedicated to recreating the contents of this paper, we also provide reference implementations of the CPD algorithms in the pip-installable Python package *changepoynt*[2]. By making the algorithms easily accessible, we aim to enable future applications to benefit from the proposed (IKA-)SST improvements.

II. RELATED WORK

This paper does not propose a new CPD algorithm. Instead, we improve an existing algorithm (SST [2]), and another variant of it (IKA-SST [3]). Therefore, we mainly discuss publications related our target algorithms, including methods based on Singular Spectrum Analysis (SSA) [6]. In addition to the related work on SST-based CPD, we discuss publications whose contributions involves the efficient decomposition of Hankel matrices, as we employ similar algorithms to improve SST-based CPD.

A. CHANGE POINT DETECTION

CPD methods aim to identify abrupt structural changes within time series. While there are different methods from statistics [7] and other domains [8]–[12], this paper focuses on techniques that employ SSA-based methods as discussed in beginning of this section. SSA-based methods extract time series characteristics by computing the SVD of matrices that contain subsequence sampled from a time series. The degree of change is estimated by comparing the extracted characteristics at two different times within the series. We refer to Section III-C for a detailed explanation on the SST. For a broader overview of time series CPD, see [1].

Moskvina et al. [6] build on the theory of SSA and compare the extracted characteristics with time series samples at a later point in time to detect changes. Idé et al. [2] published a variation of the algorithm termed SST, which they later improve for speed by adopting an implicit Krylov subspace approximation [3] (IKA-SST). These algorithms are widely accepted [1] and have been applied to different use cases [4], [5], [13]. The authors claim that IKA-SST is carefully designed to account for numerical errors caused by the subspace reduction but only show qualitative measurements to support that claim. Other, more recent improvements of the SSA-based CPD methods, mainly aim at improving the scoring function [5], [14], [15], or the application to multidimensional time series [16], rather than their computational efficiency.

Our paper builds on the related work on SSA-based change point detection, mainly addressing the runtime and computational complexity of the algorithms. While the IKA-SST is an already improved version of the SST, we improve it further. In our evaluation and discussion section (Sec. IV-E & V), we show how our improvements introduce only a small approximation error, arguing that our algorithmic modifications can be used as a drop-in replacement. Independent of our improvement, our data shows that the IKA-SST underestimates the score, so the SST remains a valid alternative.

[2]https://pypi.org/project/changepoynt/

B. EFFICIENT HANKEL MATRIX DECOMPOSITIONS

Hankel matrices and their decompositions are common in different areas of digital signal processing. While the goals in these areas may vary from those of CPD, we still want to consider their advancements. The following papers are related to our work as they share the SVD of a Hankel matrix and propose an efficient decomposition as part of their contribution. For a clearer distinction of our approach to the related work in this section, we want to reiterate that we apply two different algorithms from linear algebra to improve the SST and the IKA-SST. First, we employ the rSVD as a fast, truncated SVD. Second, we use a fast matrix product, exploiting the Hankel structure of our matrices.

Singular Spectrum Analysis. Korobeynikov [17] published an efficient decomposition of Hankel matrices with similar complexity as proposed in our paper. While their paper uses the structure of the Hankel matrix, the decomposition relies on a Lanczos bidiagonalization method, which is iterative in nature. The rSVD used in our paper supports parallelization, facilitating better utilization of modern, multicore CPUs. Still, we employ the Lanczos bidiagonalization as a baseline and show how the rSVD is comparable in accuracy, but is faster in practice. Furthermore, Korobeynikov [17] shows runtime measurements, but numerical errors and the method's applicability are neither discussed nor empirically measured. Rodrigues et al. [18] use randomized matrix decompositions to speed up SSA techniques for large time series but do not exploit the structure of the Hankel matrix, which allows a fast Hankel matrix product. Again, they mainly show time measurements and have limited to no discussion of errors.

Digital Signal Processing. Sahnoun et al. [19] use Lanczos bidiagonalization in combination with the structure of the (multilevel) Hankel matrix to arrive at a the same computational complexity as this paper. Still, the decomposition is iterative in nature, similar to [17]. Their paper also includes discussions and measurements of error, but only in the context of parameter estimation for dynamical systems, such as wireless communication channel estimation or antenna systems. Yang et al. [20] propose to use rSVD for signal denoising based on Hankel matrices. In contrast to our paper, they do not exploit the Hankel structure. Wang et al. [21] apply rSVD for large-scale system identification and also include a thorough error discussion focused on their specific application. Their application demonstrates that the Hankel matrices can quickly grow in size. In contrast to our paper, their approach does not use the Hankel structure or the parallelization inherent in the rSVD.

In general, all papers in this section differ from our research as they did not evaluate their methods in the context of SST-based time series CPD. We contribute theoretical arguments for the suitability of the rSVD in our CPD scenario and derive specialized error bounds. Furthermore, our contribution is not limited to the Hankel matrix rSVD for the SST, but we also improve the IKA-SST through the fast Hankel matrix product.

Algorithm 1 Singular Spectrum Transformation (SST) [2]. Final scoring function from [3].

```
Input: Time series $T \in \mathbb{R}^{1\times L}$, window size $N$, target rank $k$, lag $\delta$ between the Hankel matrices
Result: Series of scores $S \in \mathbb{R}^{1\times L}$
1: Initialize empty series of scores $S \in \mathbb{R}^{1\times L}$ with elements $S_i$
2: for $i = (2N - 1 + \delta)$ to $L$ do
3:     Create future Hankel matrix $H_f$ from the signal range $[i - (2N - 1),\ i]$
4:     Compute the prominent singular vector $\mathbf{u}_f \in \mathbb{R}^{N\times 1}$ of future Hankel matrix $H_f$
5:     Create past Hankel matrix $H_p$ from the signal range $[i - (2N - 1 + \delta),\ i - \delta]$
6:     Compute the $k$ most prominent singular vectors $U_p \in \mathbb{R}^{N\times k}$ using past Hankel matrix $H_p$      } Replace with alg. 2
7:     Compute the score as one minus the norm of the matrix vector product $S_i = 1 - ||U_p^T \mathbf{u}_f||_2^2$      } for IKA-SST.
8: end for
9: return $S$
```

III. METHODS

Our goal is to increase the efficiency of both the SST and IKA-SST without introducing large approximation errors. In the following sections, we explain the necessary, basic concepts (Sec. III-A & III-B) to understand the SST and IKA-SST. We introduce both algorithms, and discuss their main computational bottlenecks (Sec. III-C). Subsequently, we explore the necessary methods to tackle these bottlenecks (Sec. III-D & III-E), and then explain how we use them to improve the SST and IKA-SST, respectively (Sec. III-F & III-G). Fig. 2 visualizes the algorithmic modifications to improve the SST efficiency. On a general level, different parts of the SST (alg. 1) can either be replaced by the IKA subroutine (alg. 2), or the rSVD (alg. 3). Both contain Hankel matrix multiplications, which can be replaced by a fast Hankel matrix product (alg. 4).

In addition to the algorithmic improvements, Section III-F discusses theoretical reasons for using the rSVD as part of the SST algorithm. On a high level, a truncated decomposition like the rSVD allows us to trade accuracy for speed. We examine this trade-off to show how the rSVD is well suited as a decomposition for Hankel matrices in terms of approximation errors (Sec. III-H), and derive an error bound that directly extends this arguments to the SST score (Sec. III-I).

The notation table in Appendix A provides an overview of the symbols used in this section.

A. HANKEL MATRICES

Hankel matrices H are special matrices with constant skew diagonals (e.g., fig. 1). While they may come up differently for other applications, they can be constructed from time series by filling the columns with subsequences using a sliding window over the original time series as visualized in fig. 1. For a window size N, the Hankel matrix contains $2N - 1$ different samples overall. Note that the rows and columns represent sliding windows over the original time series, and the first row and last column together contain the complete time series used for constructing the matrix. A square Hankel matrix is symmetric, and, therefore, equal to its transpose ($H = H^T \ \forall\, H \in \mathbb{R}^{N\times N}$). Hankel matrices are closely related to Toeplitz matrices. Reversing the column order transforms a Hankel matrix into a Toeplitz matrix.

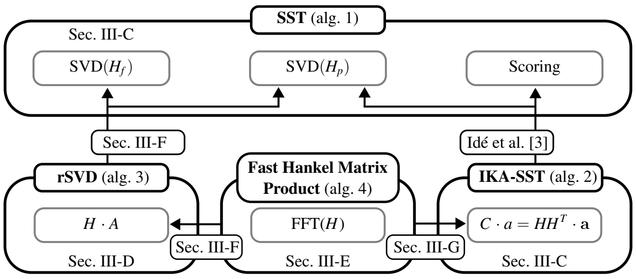


FIGURE 2: Visualization of the algorithmic interactions. Arrows indicate the replacement of an operation.

B. SINGULAR VALUE DECOMPOSITION

The SVD of a matrix $\text{SVD}(H) = U\Sigma V^T$ results in three matrices. U contains the left singular vectors in its columns, diagonal matrix Σ contains the singular values σ_i in decreasing order ($\forall i\ \sigma_i \geq \sigma_{i+1},\ i \in [1, min\{M, N\}]$), and V^T contains the right singular vectors as rows. In the case of a complete SVD, multiplying the matrices perfectly reconstructs the original matrix. The computational effort for the complete SVD depends on the matrix dimensions $\mathcal{O}(M * N * min\{M, N\})$, where M is the number of rows and N is the number of columns of the matrix [22]. For quadratic Hankel matrices with $M = N$, the complexity of the SVD is $\mathcal{O}(N^3)$.

Many applications only require the $k \ll N$ most prominent singular values and corresponding singular vectors U_k, Σ_k, and V_k. Therefore, a complete decomposition is often unnecessary. This gives rise to the development of partial decompositions that stop at k components and are referred to as truncated SVD algorithms.

We restrict the complexity terms and error bounds to $N = M$ for clarity, but all following algorithms also work for $N \neq M$.

C. SINGULAR SPECTRUM TRANSFORMATION

SSA-based CPD algorithms measure the degree of change in a time series by comparing the largest left singular vectors of two Hankel matrices. These matrices are constructed using time series samples from before and after some time point t. Intuitively, the largest singular vectors primarily capture the main characteristics of the time series used to construct the matrices, whereas the smaller ones mostly contain noise and minor variations. If these characteristics differ before and after t, a change has occurred.

In the scope of our paper, a time series is an ordered collection of tuples of length L, each consisting of a timestamp and a real signal value. We assume that the timestamps are equidistant from each other. A sliding window of length l returns l consecutive elements from the original time series.

On a high level, the SST algorithm [3] can be condensed into the steps shown in alg. 1. At every sample of a complete time series, two Hankel matrices, past and future, are constructed from the original time series T. The past Hankel matrix H_p contains time series samples before a point t, whereas the future Hankel matrix H_f contains samples after that point. Changes are detected by comparing the future Hankel matrix's most prominent left singular vector $\mathbf{u}_f$ to the k most prominent singular vectors U_p of the past Hankel matrix. Mathematically, the final score is computed as one minus the norm of the projection of $\mathbf{u}_f$ on the past singular vectors U_p ($||U_p^T \mathbf{u}_f||_2$, alg. 1, line 7).

Therefore, every SST-step requires two truncated decompositions of $N \times N$ Hankel matrices and a matrix vector product. With increasing window sizes, these decompositions quickly become the computational bottleneck of the algorithm.

The IKA-SST [3] is an improved version of the SST that aims to increase the speed of the original SST. The paper shows that the score as defined in alg. 1 can be computed in a subspace seeded by the future singular vector $\mathbf{u}_f$ without ever computing the past singular vectors U_p explicitly. The subspace is only slightly larger than the hyperparameter k, significantly smaller than N. See the steps in alg. 2 for details. This effectively removes the SVD of the large $N \times N$ Hankel matrix as a computational bottleneck. Although this dramatically reduces the runtime, the algorithm still requires the computation of the correlation matrix as a product of two matrices (alg. 2, line 1). A step that is embarrassingly parallelizable but still has cubic complexity $\mathcal{O}(N^3)$ concerning the matrix size N.

Note that we are using the updated scoring function $S_i = 1 - ||U_p^T \mathbf{u}_f||_2^2$ from the IKA-SST for both variants. The reasons are explained in the IKA-SST paper [3].

Algorithm 2 IKA-SST subroutine [3].

```
Input: Past Hankel matrix $H_p \in \mathbb{R}^{N \times N}$, target rank $k$,
    prominent future singular vector $\mathbf{u}_f \in \mathbb{R}^{N \times 1}$
Result: Approximated change point score $\hat{S}_i$
 1: Compute correlation matrix $C = H_p H_p^T \in \mathbb{R}^{N \times N}$
 2: Set Lanczos rank $\zeta = \begin{cases} 2k & k \in \text{even} \\ 2k-1 & k \in \text{odd} \end{cases}$
 3: Initialize $\alpha \in \mathbb{R}^{1 \times (\zeta+1)} = 0$, $\beta \in \mathbb{R}^{1 \times (\zeta+1)} = 1$
 4: Initialize $\mathbf{r} = \mathbf{u}_f$, $\mathbf{q} \in \mathbb{R}^{N \times 1} = 0$
 5: for $i = 1$ to $\zeta$ do                          ▷ Lanczos Subroutine
 6:     $\tilde{\mathbf{q}} = \mathbf{r}/\beta_{0,i}$
 7:     $\alpha_{0,i+1} = \tilde{\mathbf{q}}^T C \tilde{\mathbf{q}}$
 8:     $\mathbf{r} = C\tilde{\mathbf{q}} - \alpha_{0,i+1}\tilde{\mathbf{q}} - \beta_{0,i} * \mathbf{q}$
 9:     $\beta_{0,i+1} = ||\mathbf{r}||$
10:     $\mathbf{q} = \tilde{\mathbf{q}}$
11: end for
12: Tridiagonal matrix $\tilde{C} \in \mathbb{R}^{\zeta \times \zeta}$ with main diagonal set to
    $\alpha[1:]$ and off-diagonals set to $\beta[1:-1]$
13: Compute SVD$(\tilde{C}) = \tilde{U}\tilde{\Sigma}\tilde{V}^T$       ▷ Special Tridiag. SVD
14: $\hat{S}_i = 1 - \sum_{j=0}^{k-1} (\tilde{U}_{0,j})^2$   ▷ First k-elements of first row of $\tilde{U}$
15: return $\hat{S}_i$
```

D. RANDOMIZED SINGULAR VALUE DECOMPOSITION

In their seminal paper, Halko et al. [22] present a modular framework for constructing randomized algorithms for truncated matrix decompositions, one of which is the randomized SVD (rSVD). The basic idea of rSVD is to create a smaller, representative subspace of the original matrix A by randomly sampling from its column space. The original matrix A is then projected onto this smaller subspace. Computing the SVD in this subspace is significantly faster due to the reduced dimensionality. Transforming the singular vectors from the subspace to the original space yields the final result. The suitability of the subspace, represented by the subspace projector Q, directly influences the projection error. Determining a suitable subspace is the focus of the rSVD algorithm.

The rSVD algorithm proposed by Halko et al. [22] can be seen in alg. 3. Lines 1-10 contain all operations to find a suitable Q. The first step is the random sampling from the column space using a random matrix $\Omega \in \mathbb{R}^{N \times (k+p)}$ to initialize the subspace (lines 1 & 2). An oversampling parameter p is used to enhance the precision of the decomposition by sampling $k+p$ columns. Projector Q should have orthonormal columns that represent most of the original space to make the projection as good as possible:

$$QQ^T \approx I \implies A \approx QQ^T A \tag{1}$$

Additionally, subspace iterations can be used to enhance the dominant singular values. The iterations decay the contribution of lower singular values by taking repeated powers of the input matrix. These power iterations are mixed into the creation of Q in the original paper (alg. 3, lines 4-9).

Due to the projection into a subspace with fewer dimensions, the complexity of the SVD in alg. 3 reduces to $\mathcal{O}((k+p)^2 N)$ and the bottleneck shifts to the q repeated matrix multiplications of the input matrix $A \in \mathbb{R}^{N \times N}$ with other matrices $\Omega, Q \in \mathbb{R}^{N \times (k+p)}$ for the sampling step and the subspace iterations. This keeps the overall complexity quadratic with respect to window size N ($\mathcal{O}(q(k+p)N^2)$). In contrast to other Krylov subspace methods, like Lanczos bidiagonalization, used in other publications that have iterative components, the matrix multiplications required to construct the subspace projector are embarrassingly parallelizable [22].

E. FAST HANKEL MATRIX PRODUCT

While the rSVD can be applied to matrices of arbitrary structure, our matrices show Hankel structure due to the con-

Algorithm 3 Randomized Singular Value Decomposition (rSVD) with Subspace Iterations [22].

```
Input: Matrix $A \in \mathbb{R}^{N\times N}$, target rank $k$, oversampling parameter $p$, number of subspace iterations $q$
Result: Approximated, truncated SVD $(A) \approx \hat{U}_k\hat{\Sigma}_k\hat{V}_k^T$
 1: Generate a random matrix $\Omega \in \mathbb{R}^{N\times(k+p)}$.
 2: Form $Y_0 := A\Omega$                                  ▷ Random Sampling.
 3: Compute QR factorization $Y_0 := Q_0R_0$ for initialization
 4: for $i = 1$ to $q$ do                                  ▷ Subspace Iterations.
 5:     Form $\tilde{Y}_i := A^TQ_{i-1}$
 6:     Compute its QR factorization $\tilde{Y}_i = \tilde{Q}_i\tilde{R}_i$
 7:     Form $Y_i := A\tilde{Q}_i$
 8:     Compute its QR factorization $Y_i = Q_iR_i$
 9: end for
10: Save $Q = Q_i$
11: Form the matrix $B \in \mathbb{R}^{(k+p)\times N} := Q^TA$
12: Compute the SVD of the small matrix $B = \tilde{U}_k\hat{\Sigma}_k\hat{V}_k^T$
13: Project $\hat{U}_k := Q\tilde{U}_k$ back into original space
14: return $\hat{U}_k \in \mathbb{R}^{N\times k}$, $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$, and $\hat{V}_k \in \mathbb{R}^{N\times k}$
```

struction with sliding-window subsequences. Every column and row of the Hankel matrix is a sliding window of size N. Therefore, multiplying the matrix by a vector from the left or right is equivalent to the cross-correlation of the vector with the larger window of size $2N-1$. The naive matrix vector product of vector $\mathbf{a} \in \mathbb{R}^{N\times 1}$ with an arbitrary matrix $A \in \mathbb{R}^{N\times N}$ has a computational complexity of $\mathcal{O}(N^2)$.
Convolutions, particularly for larger kernels, can be efficiently computed in the frequency domain as noted in the convolution theorem [23]. The theorem states that the convolution in the time domain is a multiplication in the frequency domain. To convert the discrete cross-correlation into a discrete convolution, the vector has to be flipped (reverse element order). Due to the availability of the Fast Fourier Transform (FFT), the complexity of the multiplication of a Hankel matrix $H \in \mathbb{R}^{N\times N}$ with a vector $\mathbf{a} \in \mathbb{R}^{N\times 1}$ reduces to $\mathcal{O}(N\log N)$. Multiplying the Hankel matrix by any arbitrary matrix $A \in \mathbb{R}^{N\times(k+p)}$ can be reduced to multiple matrix vector products resulting in a complexity of $\mathcal{O}((k+p)N\log N)$. The multiplication of H by each of the $k+p$ columns of A is an independent operation and can therefore run in parallel.
Additionally, there is no need to construct the full Hankel matrix and allocate $\mathcal{O}(N^2)$ memory, which costs additional time and may cause memory problems for large window sizes. It is only ever necessary to compute the FFT for sliding windows of size $2N-1$, reducing the space complexity of the algorithm to a linear function of the window size $\mathcal{O}(N)$.

Alg. 4 lists the necessary steps to perform an efficient, aperiodic cross-correlation using the FFT. To multiply a Hankel matrix by an arbitrary matrix A, the algorithm basically performs several matrix vector products. The multiplication algorithm is not new and has been published several times before [17]. The algorithm iterates over the columns of

Algorithm 4 Fast Hankel Matrix Multiplication hankel_matmul.

```
Input: FFT representation of Hankel matrix $H \in \mathbb{C}^{1\times(2N-1)}$, arbitrary matrix $A \in \mathbb{R}^{N\times G}$ with columns $A_i$
Result: Multiplication result $R \in \mathbb{R}^{N\times G}$
 1: function hankel_matmul($H, A$)
 2:     Create empty matrix $R$ with columns $R_i$
 3:     for $i = 1$ to $G$ do                 ▷ Iterations can run in parallel.
 4:         Flip the column $\hat{A}_i = A_i[:: -1]$ (and zero-pad)
 5:         FFT $\tilde{A}_i = \mathscr{F}(\hat{A}_i)$     ▷ padding: Sec. III-E & III-F
 6:         Multiplication $\tilde{R}_i = \tilde{A}_iH$                ▷ element-wise
 7:         Inverse FFT $\hat{R}_i = \mathscr{F}^{-1}(\tilde{R}_i)$
 8:         Extract column $R_i = \hat{R}_i[(N-1) : (2N-1)]$
 9:     end for
10:     return $R$
11: end function
```

arbitrary matrix A, and performs the same, independent set of operations on each column of A. Beginning in line 4, we flip the column to account for the difference in cross-correlation and convolution, and zero-pad the column to the same size as the FFT-transformed Hankel matrix. We then perform the FFT on the column (line 5), and perform the multiplication in frequency space (line 6). After that, we apply the inverse FFT transform (line 7) and extract the elements from $N-1$ to $2N-1$ into the corresponding column of result matrix R to avoid circular convolution (line 8) [24].
The fast Hankel matrix product also applies to matrices that introduce a lag τ between the sliding time windows. In this case, the resulting matrix would not be a Hankel matrix by definition. However, we can account for these gaps. We do this by introducing τ zero-filled rows between all original rows of A, thereby increasing the number of rows from N to $N+(N-1)\tau$. The length of the time series in the Hankel matrix increases to $(2N-1)+(N-1)\tau$. In this case, the complexity for the matrix product results to $\mathcal{O}((k+p)\tau N\log(\tau N))$. The matrix vector product is, therefore, also applicable to matrices that do not strictly have Hankel structure but are similar in that the columns are subsequences of a time series. The speedup using the convolution theorem then depends on the value of τ.

F. IMPROVING SST THROUGH THE RSVD
We have now introduced all the necessary building blocks that we will use to improve the speed of the SST and IKA-SST. In short, we improve the SST (alg. 1) by replacing the SVD with the rSVD (alg. 3), and the Hankel matrix multiplications in the rSVD with the fast Hankel matrix product (alg. 4).
If the input matrix A to the rSVD (alg. 3) is a Hankel matrix, we can replace every product involving A with the fast Hankel matrix product (alg. 4). For preparation, we only have to transform the Hankel matrix into frequency space by applying the FFT to the entire time series of length $2N-1$

used to construct the Hankel matrix (see Sec. III-A). The replacement of the naive matrix product directly addresses the bottleneck of the rSVD. The overall complexity of the rSVD then decreases from $\mathcal{O}(q(k+p)N^2)$ to $\mathcal{O}(q(k+p)N\log N)$. Alg. 5 (FFT rSVD) lists all steps of the specialized rSVD with detailed comments on the complexity of each operation. The hyperparameter k of the SST directly translates to the target rank k of the rSVD.

In addition to the main algorithmic adaptations discussed in the previous sections, we made several minor adaptations to the original algorithm. We substitute the QR-Factorization with the LU-Factorization as done in Li et al. [25] to further improve the runtime. Additionally, we zero-pad the original signal and use an FFT algorithm for real inputs. We zero-pad the vectors to reach an optimal length $\geq 2N-1$ for the divide and conquer FFT algorithm (provided by the FFT implementation).

The rSVD with the fast Hankel matrix product yields the following advantages over other decomposition methods:

- **Parallelization.** The term $(k+p)$, introduced to the computational complexity by the oversampling, is parallelizable over multiple cores in contrast to other, iterative methods. Improving the accuracy using the oversampling parameter p is not as costly in terms of processing time, as it can be offset by adding more workers.
- **Modifiable Error Bounds.** The upper error bounds are configurable by adapting the oversampling parameter p and the number of subspace iterations q.
- **Concise Implementation**. The implementation of the rSVD is straightforward, involving only a few steps and very common matrix operations. These are most likely readily available even on specialized hardware, such as GPUs and microprocessors.
- **Computational and Space Efficiency.** The computational complexity of the decomposition is only $\mathcal{O}(q(k+p)N\log N)$. Implemented in code, this yields significant speedup for the truncated decomposition of Hankel matrices compared with the complete SVD. Additionally, the space complexity has been reduced to $\mathcal{O}((k+p)N)$ from $\mathcal{O}(N^2)$, because there is no need to construct the full Hankel matrix ($k,p,q \ll N$).

As we replace the SVD in the SST algorithm (FFT rSVD SST) with the specialized rSVD (FFT rSVD), all advantages of the rSVD transfer to the SST as well. In addition to the listed advantages, we argue that the rSVD will achieve low approximation errors for the Hankel matrices in our scenario. See Sections III-H and III-I for further details.

The standard SVD algorithm [26] overwrites the input matrix structure to compute all components as accurately as possible. Therefore, we can not improve the complete SVD using the fast Hankel matrix product to replace the multiplications with the input matrix. Additionally, we only need the first few components, which is why we targeted truncated decompositions. Note that our improvement of the rSVD is not limited to perfect Hankel matrices (lag of one), but only requires that the rows/columns are subsequences of one larger time series, sampled with some delay τ (see the end of Sec. III-E). Furthermore, this specialized rSVD version also applies to all matrices that are a product of Hankel matrices owing to the associativity of the multiplication. Observe that the input matrix A to the rSVD is only ever used in multiplications, so we never have to actually compute the product of these Hankel matrices. We will use a similar observation for improving the IKA-SST in the next section.

G. IMPROVING THE IKA-SST

For the IKA-SST, the implicit Krylov approximation shifted the bottleneck from the SVD to computing the correlation matrix C in line one of alg. 2. This multiplication is necessary to maintain a low approximation error, but has a complexity of $\mathcal{O}(N^3)$, scaling badly with the window size. To understand how to improve the IKA-SST, observe that the matrix C is only ever used directly in matrix vector multiplications (alg. 2, lines 7 & 8). By exploiting the associativity of the multiplication, we never have to compute C explicitly. We can skip line one and replace the matrix products (lines 7 & 8) with two fast Hankel matrix products $C\tilde{\mathbf{q}} = (HH^T)\tilde{\mathbf{q}} = H(H^T\tilde{\mathbf{q}})$. The product in brackets is a matrix vector product and the same is true for multiplying H by the resulting vector $H^T\tilde{\mathbf{q}}$ of the previous operation. Thus, we eliminate the computational bottleneck of the IKA-SST. The following sections will refer to this modified algorithm as FFT IKA-SST. Similar to the rSVD and SST, the space complexity reduces to $\mathcal{O}(N)$. This concludes our second objective of improving the IKA-SST. For readers interested mainly in the practical implications, we recommend jumping to Section IV-F ff, where we discuss the speedup.

H. DISCUSSION OF DECOMPOSITION ERRORS

Now that we have introduced our improvements to the SST and the IKA-SST, we want to provide further reasons for choosing the rSVD instead of other truncated decompositions, such as the implicitly restarted Lanczos bidiagonalization (IRLB) [27].

Summarized in Section III-F and discussed in detail by Halko et al. [22], the rSVD has several advantages over its competitors. As the main disadvantage they mention that, with the same computational budget, the rSVD is in many cases less accurate than, e.g., the IRLB. Nevertheless, we argue that the rSVD is comparably accurate to other sequential competitors in our special use case. To substantiate this claim, we discuss theoretical error bounds and how the spectral properties of time series Hankel matrices align conveniently with requirements derived from these bounds. Before doing so, we have to introduce general error bounds for both the truncated SVD in general and the rSVD in detail.

Minimal Truncation Error. Using all singular vectors and singular values computed by the SVD perfectly reconstructs the initial matrix. Truncation of the computed spectrum always introduces an error to the reconstruction. The absolute error depends directly on the power of the truncated (unused) singular values. The Eckart-Young-Mirsky theorem [28], [29]

Algorithm 5 Efficient truncated Hankel Decomposition based on rSVD.

```
Input: Time series $T \in \mathbb{R}^{1\times L}$, window size $N$, target rank $k$, oversampling parameter $p$, number of subspace iterations $q$
Result: Approximated, truncated Hankel SVD $(H) \approx \hat{U}_k\hat{\Sigma}_k\hat{V}_k^T$
1: function hankel_rsvd($T, N, k, p, q$)
2:     // Create the Hankel representation (with padding to next best length $\geq 2N-1$)
3:     Extract the flattened Hankel matrix as a sliding window $\tilde{H} \in \mathbb{R}^{1\times(2N-1)}$ from $T$      ▷ $\mathcal{O}(N)$
4:     Compute the FFT $H := \mathscr{F}(\tilde{H})$ representation of the Hankel matrix      ▷ $\mathcal{O}(N\log N)$

5:     // Randomized sampling from column space
6:     Generate a random matrix $\Omega \in \mathbb{R}^{N\times(k+p)}$ with i.i.d. entries from a standard normal distribution.      ▷ $\mathcal{O}(N(k+p))$
7:     Form $Y_0 :=$ hankel_matmul$(H, \Omega)$      ▷ $\mathcal{O}((k+p)N\log N)$

8:     // Begin of Subspace iterations with $Q \in \mathbb{R}^{N\times(k+p)}$
9:     Compute QR factorization $Y_0 = Q_0R_0$ for initialization      ▷ if $q > 1$: LU factorization [25].
10:    for $i = 1$ to $q$ do      ▷ Complete loop: $\mathcal{O}(q(k+p)N\log N)$
11:        Form $\tilde{Y}_i :=$ hankel_matmul$(H, Q_{i-1})$      ▷ alg. 4, $H^T = H$ (Sec. III-A)
12:        Compute its QR factorization $\tilde{Y}_i = \tilde{Q}_i\tilde{R}_i$      ▷ while $i < q$: LU factorization [25].
13:        Form $Y_i :=$ hankel_matmul$(H, \tilde{Q}_i)$      ▷ alg. 4
14:        Compute its QR factorization $Y_i = Q_iR_i$      ▷ while $i < q$: LU factorization [25].
15:    end for

16:    // SVD in Subspace
17:    Form the matrix $B \in \mathbb{R}^{(k+p)\times N} :=$ (hankel_matmul$(H, Q_q))^T$      ▷ $\mathcal{O}((k+p)N\log N)$, $Q_q^TH = (HQ_q)^T$, with $H^T = H$
18:    Compute the SVD of the small matrix $B = \tilde{U}_k\hat{\Sigma}_k\hat{V}_k^T$      ▷ $\mathcal{O}((k+p)^2N)$
19:    Project $\hat{U}_k := Q_q\tilde{U}_k$ back into original space      ▷ $\mathcal{O}(N(k+p)^2)$
20:    return $\hat{U}_k \in \mathbb{R}^{N\times k}$, $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$, and $\hat{V}_k \in \mathbb{R}^{N\times k}$
21: end function
```

states that a rank k reconstruction A_k using the first k singular vectors is better or equal to any other matrix with rank k in terms of the Frobenius $(||\bullet||_F)$ and spectral l_2 norm $(||\bullet||_2)$ (for a decreasingly sorted singular spectrum). The error of the truncated SVD reconstruction of rank k can be computed as follows [28], [29]:

$$||A - A_k|| = \begin{cases} \sigma_{k+1} & \text{for the } ||\bullet||_2 \text{ norm} \\ \left(\sum_{i=k+1}^{r}\sigma_i^2\right)^{1/2} & \text{for the } ||\bullet||_F \text{ norm} \end{cases} \tag{2}$$

$$||A - X|| \geq ||A - A_k|| \text{ for all } X \text{ with rank}(X) = k \tag{3}$$

In other words, the error caused by the truncation is directly related to the singular value spectrum of the matrix A and the decay therein. The faster the spectrum decays, the better is the truncated reconstruction. For our analysis of errors in this section, we will concentrate on the $l2$ norm similar to Halko et al. [22].

rSVD Approximation Error. The rSVD relies on random sampling, so the error bounds can only be formulated as an expected error $\mathbb{E}^*$ over the realizations of the sampling step. The error is introduced mainly by the subspace projection in line 11 of alg. 3. Halko et al. [22, Corollary 10.10] arrive at the following error bounds for this step (with Euler's number e, target rank k, oversampling p, subspace iterations q, and matrix size N):

$$\mathbb{E}^*_{\text{Proj}} := \mathbb{E}||A - QQ^TA||_2 \tag{4}$$

$$\mathbb{E}^*_{\text{Proj}} \leq \underbrace{\left[1 + \sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p}\sqrt{N-k}\right]^{\frac{1}{2q+1}}}_{=:f^k_{proj}} \sigma_{k+1} \tag{5}$$

$$\mathbb{E}^*_{\text{rSVD}} \leq \sigma_{k+1} + \mathbb{E}^*_{\text{Proj}} \tag{6}$$

$$\text{with q} \sim \log N: \ \mathbb{E}^*_{\text{Proj}} \sim \sigma_{k+1} \implies \mathbb{E}^*_{\text{rSVD}} \sim 2\sigma_{k+1} \tag{7}$$

Note that we replaced the tail term $(\sum_{j>k}\sigma_j^{2(2q+1)})^{1/2}$ with its largest term $\sqrt{N-k}\cdot\sigma^{2q+1}$ as done in the original paper [22, Corollary 10.10 f]. This is pessimistic but makes clear how the truncation error of the rSVD is connected to the SVD. When comparing (6) with (2), we see that the rSVD introduces an additive error to the minimal possible value of σ_{k+1}. This additive term quantifies the fidelity of the subspace projection and decreases with an increasing oversampling parameter p and even exponentially with an increasing number of subspace iterations q. When q approaches the vicinity of $\log N$, the additional error is only the largest dropped singular value [22]. In addition, we should note that the size of the matrix N increases the error. The upper bound is described as overly pessimistic in the original publication but gives an analytical expression that shows the essential factors for the method's applicability. The equation shows that the subspace projection works best for matrices with a rapidly decaying singular spectrum and, therefore, a small σ_{k+1}. Consequently, replacing the computationally expensive SVD with the truncated rSVD works best for matrices with this property.

General Hankel Matrices. Intuitively, Hankel matrices from non-chaotic time series should have a fast-decaying singular spectrum, because the columns are constructed using a sliding time window over the time series. Since the windows largely overlap, it is likely that the columns are highly correlated, and the resulting Hankel matrix is of low rank. In other words, only a few vectors are necessary to reconstruct the matrix. This intuition is formalized in [30]. The paper notes that linear recurrence relations in the time series are a necessary and sufficient condition for rank-deficient Hankel matrices. Based on these findings, we argue that the rSVD is suitable for the truncated decomposition of Hankel matrices.

SST-based Hankel Matrices. When using the rSVD as part of the SST, the low-rank assumption becomes a necessary precondition for even applying the SST algorithm. The SST, an SSA-based algorithm [2, Sec. 5], is mostly limited to sufficiently structured signals, resulting in low-rank Hankel matrices [6, Sec. 6.2]. In cases where time series have stochastic or chaotic patterns, even the naive SST algorithm is not expected to deliver good detection results. We argue that, in these cases, where the naive SST is unsuitable, additional decomposition errors are not the primary concern. Consequently, we can expect low-rank Hankel matrices when replacing the SVD with the rSVD in the SST algorithm.

Positive Semi-Definite Hankel Matrices. As noted in the related work Section II, the decomposition of Hankel matrices is a common part of many algorithms for time series and system analysis. Therefore, we want to add some notes on positive semi-definite Hankel matrices. We cannot guarantee this property in our scenario, but our insights may be of interest to others. See Appendix D for further details.

I. CONNECTING DECOMPOSITION AND SCORING ERROR

From the previous section, it is clear that the rSVD is a suitable candidate for decomposing (SST-based) Hankel matrices. Nevertheless, it remains unclear how exactly the decomposition error influences the final SST score (Sec. III-C). The score depends mainly on the extracted singular vectors (alg. 1, line 7), which are not part of the error bounds in the previous section. Intuitively, an accurate decomposition yields accurate singular vectors, which in turn yield an accurate score, however, the exact mechanism remains unclear. Although our evaluation will show that this intuition is correct, this section analyzes this connection in detail. The goal of this section is to derive an analytical error bound that directly connects the rSVD decomposition error to the SST scoring error.

By interpreting the projection residual $||(A - QQ^TA)||$ (see (4)) as an effective perturbation $||E||_2$ to an arbitrary SVD-input matrix A, we can use matrix perturbation theory [31] to derive a bound for the scoring error of the SST, analytically connecting the decomposition error of the rSVD with the scoring error. At a high level, matrix perturbation theory, specifically Wedin's theorem [32], quantifies how perturbations of A propagate to its singular values and associated singular vectors. We define the scoring error E_S as follows:

$$\begin{aligned} E_S := &S_i - \hat{S}_i = (1 - ||U_p^T \mathbf{u}_f||_2^2) - (1 - ||\hat{U}_p^T \hat{\mathbf{u}}_f||_2^2) \\ = &||\hat{U}_p^T \hat{\mathbf{u}}_f||_2^2 - ||U_p^T \mathbf{u}_f||_2^2 \end{aligned} \tag{8}$$

After some modifications to the equation and relying on Wedin's theorem [32], we arrive at an upper bound on the scoring error that determined by the perturbation, i.e. the quality of the subspace projection. When replacing the SVD with the rSVD in the SST, there are two sources of error. The errors are introduced by the decomposition of H_p (E_p) and H_f (E_f), respectively. For the derivation, refer to Appendix B:

$$|E_S| \leq 4\frac{||E_p||_2}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{||E_f||_2}{\sigma_{f,1} - \sigma_{f,2}} \tag{9}$$

This bound can be used for any alternative decomposition. In our case, the perturbation $||E||_2$ is the projection error $\mathbb{E}^*_{\text{Proj}} = ||A - QQ^TA||_2$ of the rSVD written in (4). As this error is an expected value, our scoring error also transforms into an upper bound of the expected scoring error. Finally, we can tie the scoring error to the rSVD decomposition error:

$$\mathbb{E}\left[|E_S|\right] \leq 4\frac{\mathbb{E}^*_{\text{Proj},p}(k,p,q,N)}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{\mathbb{E}^*_{\text{Proj},f}(p,q,N)}{\sigma_{f,1} - \sigma_{f,2}} \tag{10}$$

Thus, a smaller decomposition error directly yields a smaller expected scoring error. By inserting (4) into (10), we see that increasing the parameter p and q also reduces the upper bound on the expected scoring error in the same way they reduce the decomposition error. Using the approximation $q \sim \log N \rightarrow \mathbb{E}^*_{\text{Proj}} \sim \sigma_{k+1}$ [22], the bound reduces to:

$$\mathbb{E}\left[|E_S|\right] \lesssim 4\frac{\sigma_{p,k+1}}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{\sigma_{f,2}}{\sigma_{f,1} - \sigma_{f,2}} \tag{11}$$

Equation (9) is independent of the decomposition method, but we use the rSVD with a Gaussian random projector, allowing us to employ specialized bounds for randomized decompositions that bound the error with high probability (derivation in Appendix C):

$$\begin{aligned} |E_S| \leq\ & 2\frac{\sigma_{f,2}}{\sigma_{f,1}} \frac{\left(\frac{\sigma_{f,2}}{\sigma_{f,1}}\right)^{2q}}{1 - \frac{\sigma_{f,2}}{\sigma_{f,1}}} \left(\frac{1}{\sqrt{p-1}} + \frac{e\sqrt{(1+p)(N-1)}}{p}\right) \\ &+2\frac{\sigma_{p,k+1}}{\sigma_{p,1}} \frac{\left(\frac{\sigma_{p,k+1}}{\sigma_{p,k}}\right)^{2q}}{1 - \frac{\sigma_{p,k+1}}{\sigma_{p,k}}} \left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p}\right) \end{aligned} \tag{12}$$

Both, (10) and (12), demonstrate how a fast-decaying singular spectrum, enlarging $\sigma_k - \sigma_{k+1}$ and decreasing σ_{k+1}, directly translates to a low expected scoring error. Equation (10) shows that the scoring error is a sum of the decomposition errors for the past- and future Hankel matrices scaled by the singular value gap $\sigma_j - \sigma_{j+1}$.

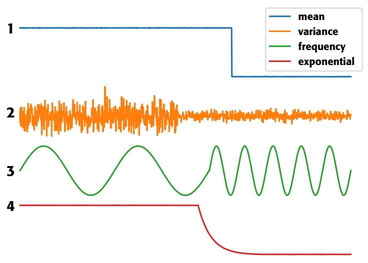


FIGURE 3: Examples for each type of simulated signals with randomized change points.

Note that the bounds are conservative and pessimistic. In practice, when the leading singular values are very close to each other (i.e., the Hankel matrix does not have a clear dominant direction, sometimes the case stochastic or weakly structured signals), the spectral gap is small and the theoretical bound on the score error becomes weak, because the bound depends inversely on the spectral gaps $(\sigma_j - \sigma_{j+1})$.

With our bound, we mathematically prove that a fast-decaying singular spectrum of the Hankel matrix leads to a low scoring error, when we replace the SVD with the rSVD in the SST algorithm. Fortunately, we expect the Hankel matrices to exhibit a fast-decaying singular spectrum. This spectral property is likely for general Hankel matrices, and even a precondition for the Hankel matrices occurring in the SST (Sec. III-H). This concludes our theoretically grounded argument on the suitability of the rSVD for SST-based CPD: The rSVD drastically improves the computational efficiency of the SST at the cost of only a small scoring error.

IV. EVALUATION

In the methods Section III, we discussed the specialized rSVD and the fast Hankel matrix product, and then used these two building blocks to improve the SST and the IKA-SST. During the discussion of the error, we showed that a fast-decaying singular spectrum results in a small approximation error (see (4)) and also directly in a small scoring error (see (10)). This theoretical evidence supports the applicability and suitability of the rSVD for Hankel matrices. Now we want to substantiate the claims with empirical measurements using a collection of simulated and real world time series. We empirically measure the singular value spectrum, decomposition quality, and scoring error. Last but not least, we measure algorithmic runtime to evaluate whether we reached our goal of speeding up SST-based CPD. Owing to the variety of measurements and amount of data, the total computation time for all experiments was approximately ten days.

Our goal is to accelerate SST-style scoring to enable scalable CPD, not to tune detectors for specific datasets. Threshold dependent metrics (e.g., F1) vary with application-specific post-processing and beyond scope of our evaluation. See the last part of the discussion Section V for further comments.

A. DATASETS

We rely on two different datasets, with real and simulated time series for our evaluation.

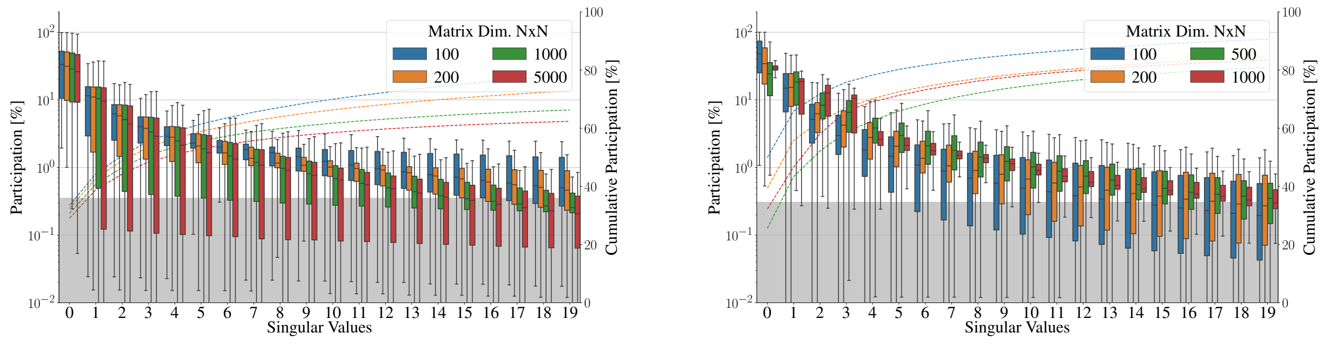
Synthetic Signals. We simulate four different types of changes. Examples of each type are shown in figure 3. Each type has two random parameters, the first one being the position of the change and the second one being the magnitude of the change. For the mean change, the magnitude is the difference between the two plateaus, and both are chosen to be a random integer percentage between zero and one. The second type is a change in the variance of two normal distributions, where the variances are chosen in a similar manner to the mean change. The third one is a change in the frequency of a sine wave with two frequencies chosen similar to the mean and variance. The fourth is an exponential decline, in which the slope of the decline is chosen randomly. Our defined types of changes are in line with the related work on CPD [3], [7]. We create this dataset to ensure that our measurements contain time series with change points.

Real Signals. For our real-world time series, we rely on the diverse UCR Time Series Classification Archive [33]. At the time of this publication, the dataset contains 191,158 signals of varying lengths from 256 different datasets. We excluded 218 signals from the dataset due to missing values.

B. IMPLEMENTATION DETAILS

The methods are implemented in Python 3.11 using linear algebra operations provided by Numpy (v1.26) [34]. Additionally, we use Numba (v0.59) [35] as a JIT compiler and for parallelization. Generally, all mathematical operations used are common. Therefore, an implementation on specialized hardware like GPUs and microprocessors is straightforward. While error measurements were run on both the Intel i9-10980XE and AMD EPYC 7713 processor, the timing measurements were performed only on the AMD processor. Both systems were running Ubuntu 24.04 with 64 GiB and 512 GiB main memory (DDR4@3200 MHz RAM) for the Intel and AMD processors, respectively. The implementation is optimized on a feasible level, but specialized lower-level languages may further reduce the runtime of our algorithm. However, the purpose of the measurements in this paper is to show the relative advantages of the methods, more than optimizing our code for the fastest absolute runtime.

Typically, the SST operates as a sliding window over the signals, processing largely overlapping time series windows. This implies highly correlated results for the scoring error measurements. To counteract this, we compute our results on non-overlapping parts of the signals. For the real-world dataset, we cut the signals in non-overlapping sections to construct the Hankel matrices. Consequently, the number of evaluated matrices decreases as the window size grows. For example, a signal with 400 samples is cut into two separate matrices for a window size of $N = 100$ (note that the entire Hankel matrix spans a larger window of size $2N - 1$). For the synthetic signals, we simulate each signal with the exact length required to construct one Hankel matrix.



(a) Synthetic Signals.

(b) Real Signals.

FIGURE 4: The singular spectrum of synthetic and real signals for different window sizes N. The values are normalized by the sum of all singular values from each matrix. The secondary axis on the right shows the cumulative participation. The gray region in the background marks the threshold for unknown noise proposed by Gavish et al. [36] around which we cut off the plot. A fast spectral decay benefits rSVD approximation error.

TABLE 1: Matrix sizes used for singular value spectrum analysis.

Synthetic Signals	100	200	1000	5000
Real Signals	100	200	500	1000

C. SINGULAR VALUE SPECTRUM

The first research question we are trying to answer is whether the rSVD is an appropriate method for the decomposition of Hankel matrices constructed from time series. The error bounds in Section III-H are mainly determined by the first excluded singular value in the truncated decomposition, as shown in (6). Therefore, we compute the complete singular value spectrum of all non-overlapping Hankel matrices extracted from our datasets for different matrix sizes (see tab. 1). For the synthetic dataset, we simulated 500 different Hankel matrices for each window size and four different signal types, resulting in 2000 singular values per boxplot, shown in fig. 4. For the real signals, we chose 1000 as the largest matrix size, because most signals in the UCR archive were shorter than 2000 samples. From the real dataset, we gathered 303057 matrices for the smallest window and 2535 for the largest window. The signals have different value ranges. Because the value range directly influences the absolute magnitudes of the singular values, we normalize each singular value by the sum of all singular values of the complete decomposition.
The resulting singular value spectra are visualized in fig. 4. Each color corresponds to a different matrix size. Each boxplot shows the distribution of the nth singular value. In addition to the boxplots, dashed lines show the cumulative percentage of singular values in the sum of all singular values. The results show an evident exponential decay of the singular value spectrum. This empirically confirms the suitability of the rSVD for Hankel matrices. Consequently, we also expect a low decomposition and scoring error.

D. DECOMPOSITION ERROR

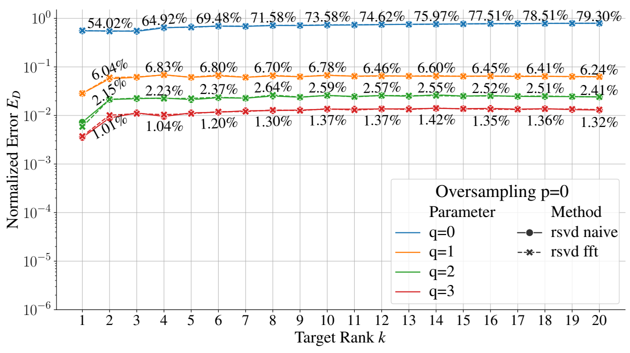
To quantify the decomposition error E_D (see (13)), we compare the reconstructed matrix with the original matrix. As an optimal baseline reconstruction, we use the singular vectors computed by the complete SVD. We compare this baseline with the reconstruction error of the singular vectors from both the rSVD and the rSVD using the fast matrix product (FFT rSVD). E_D is the difference between the reconstruction errors by the SVD and rSVD, and normalized by the SVD reconstruction error to make the results comparable for varying signal value ranges:

$$E_D(k,p,q) := \frac{\left|\left|A - A_k^{\mathrm{rSVD}(p,\,q)}\right|\right|_F - \left|\left|A - A_k^{\mathrm{SVD}}\right|\right|_F}{\left|\left|A - A_k^{\mathrm{SVD}}\right|\right|_F} \quad (13)$$

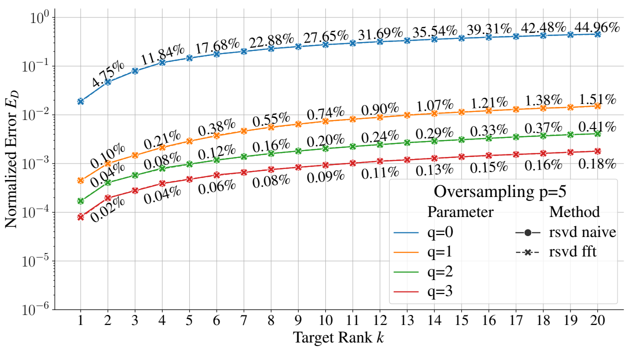
The matrix sizes are the same as those in the previous section (see tab. 1). Apart from varying the target rank k, we computed the results for different oversampling parameters p and various numbers of subspace iterations q.
The results in fig. 5 behave as expected when compared with the theoretical bounds discussed in the previous section on the rSVD error (see (4) ff.). In general, the error introduced by the rSVD is fairly small, underlining the suitability of the rSVD for Hankel matrices (Sec. III-H). Oversampling achieves good improvements. This is interesting, because the additional operations for the oversampling can be offset to some degree by parallelization. The computationally expensive subspace iterations yield the highest improvement in terms of decomposition error. Comparing the dashed and solid lines in fig. 5, we see that the fast Hankel matrix product introduces no significant error.

E. SCORING ERROR

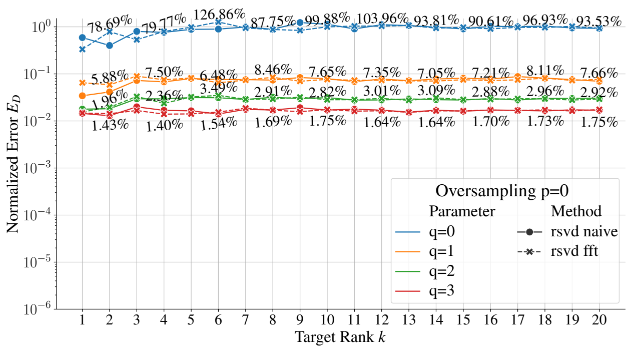
After evaluating the decomposition error, we now turn to the scoring error for the SST. The scoring is different from the reconstruction of the original matrix, as it compares the left singular vectors of two different matrices. To compute the error, we compare the different algorithms with the naive



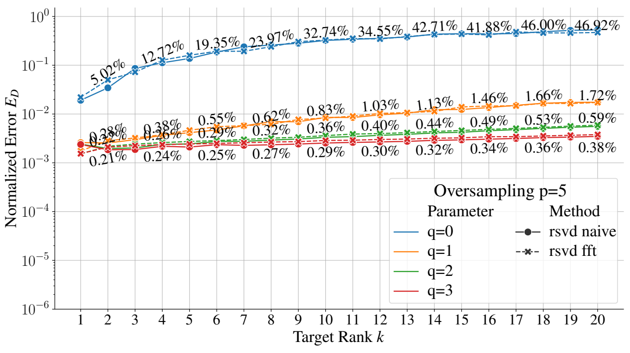
(a) Synthetic signals with oversampling $p = 0$.



(b) Synthetic signals with oversampling $p = 5$.



(c) Real signals with oversampling $p = 0$.



(d) Real signals with oversampling $p = 5$.

FIGURE 5: Decomposition error E_D (see (13)) for different target ranks k. Lower is better. The error is plotted on a logarithmic axis. Compare left and right plots for different oversampling parameters p. Compare upper and lower plots for different datasets.

implementation that uses the complete SVD for both decompositions (alg. 1, lines 4 & 6) as a baseline. Additionally, we use the implicitly restarted Lanczos bidiagonalization (IRLB) by Baglama and Reichel [27] for comparison. The IRLB is a state-of-the-art truncated decomposition with a computational complexity similar to the rSVD. Other than the rSVD it is sequential, and harder to stabilize numerically [22]. We implemented it as an alternative for computing the truncated decomposition. The IRLB implementation has been modified from an open-source repository [37]. The scoring error E_S for each method is computed as the difference between the exact score S_i and the approximated score $\hat{S}_i$ (see (8)).
Without the fast Hankel matrix product, we compare three ways to speed up the baseline algorithm:

- The IKA-SST proposed by Idé et al. [3] (IKA).
- The SST with the IRLB for the decompositions (IRLB).
- The SST with the rSVD for the decomposition (rSVD).

Each of the three variants can use the fast Hankel matrix product (FFT). Therefore, we finally evaluate six different SST variants, and the naive SVD baseline. Tab. 2 provides an overview of the computational complexities of all variants including the baseline.

Selection of Hyperparameters. Several hyperparameters must be specified for each of the variants. For all variants, we set the target rank to $k = 5$. The value is a hyperparameter from the SST and IKA-SST that does not originate from our algorithmic adaptions. SSA-based CPD papers set k to values below ten [2], [3], [6]. As a guideline they mention that k should be small enough to capture all important components, but large enough to ignore minor variations. They also mention that it should be based on the singular value spectrum, setting k so that the corresponding singular values are relatively large. As we use the rSVD, k additionally influences the computational effort and the approximation error (tab. 2 & (12)). The empirical singular value spectrum (Sec. IV-C) shows that for $k = 5$ the participation drops below 5% for all signals and window sizes reliably (within one standard deviation), with a cumulative participation well over 50%. Therefore, we argue that a value of $k = 5$ is suitable for capturing the main structure and small enough to only consider main components, but it remains a hyperparameter of the SST. This value is within the range of values used in the original publications [2], [3], [6]. Furthermore, all methods contain parameter k with the same meaning, allowing relative comparisons of the methods with a fixed k. We evaluated the effect of k on the decomposition in the previous section, and our bound shows that the effect translates to the score in a similar way (Sec. III-I).
For the IRLB, we have to specify the estimation tolerance and a maximum number of iterations. We keep these parameters to the default values of 10^{-4} for the estimation tolerance and a maximum of 50 iterations. One could further decrease

the scoring error of the IRLB with these parameters, but this would increase the runtime and the results show that the IRLB is already slower than the IKA-SST and rSVD SST with these parameters, while not being significantly more accurate.
The rSVD also requires two additional parameters: the oversampling value p and the amount of subspace iterations q. These parameters directly affect the expected error (see (6)). Based on the recommendation by Halko et al. [22], we set the number of subspace iterations to $q \approx \log N \approx 3$, as the maximum window sizes for our measurement are around 10^3 (see (10) & (12)).
Next is the oversampling parameter p. For the evaluation in this paper, we used three different orientation methods. We implemented two rank identification models for parametric time series models reviewed by Golyandina [38], namely the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). In addition, we computed the threshold for signal denoising from Hankel matrices proposed in the seminal paper of Gavish et al. [36]. To find an appropriate $k+p$, we computed the three thresholds over all matrices used in the previous section (Sec. IV-C). The mean values for all three rank estimation methods are listed in tab. 3. Based on this empirical observation, we set the oversampling parameter p so that $k+p = 15$. Our reasoning is that all further singular values after 15 will likely contain only noise, which is irrelevant for shape-based CPD.
Results. Similar to Sections IV-C and IV-D, we compute the scoring error on matrices of different sizes extracted from both datasets. In contrast to the previous evaluation, two different matrices have to be extracted before and after a certain point t (see alg. 1). For the synthetic dataset, we created 72000 matrix pairs spread over 30 different window sizes on a log scale ranging from 100 to 5000. From the real signals, we extracted 5.498.270 matrix pairs spread over 83 different window sizes on a log scale ranging from 100 to 1169 for each decomposition method. The mean absolute errors are consolidate in tab. 4. Their distribution is shown in fig. 6.
Tab. 4 clearly shows that using the fast Hankel matrix product (FFT) introduces no significant scoring error. Overall, the accuracy of our rSVD SST is comparable to the IRLB SST and slightly better than the IKA-SST for the given parametrization. Tab. 4 also shows that the scoring error is notably different between real and synthetic signals. We attribute this difference to the distribution of change scores between the two datasets. While the synthetic data aims to introduce deliberate changes via the simulation, the real signals show no changes more frequently.
In the histogram (fig. 6), the most notable difference between the methods is the high frequency of positive change score errors when using the IKA-SST (on the right tail). A positive error implies that the baseline detects a high change score, while the approximated method reports a lower score. In other words, "there are a few peaks which are not reproduced by IKA" [3, Sec. 5.2]. Our results empirically reproduce the qualitative statement in the original IKA-SST paper on a larger dataset as a by-product of our evaluations. Comparing the FFT IKA-SST with the IKA-SST in tab. 4 and fig. 6, the results also show that this difference originates from the original algorithms, and not from our modifications.
The difference in IKA-SST and SST scoring is not directly within the scope of our evaluation, because the methods are proposed elsewhere [2], [3]. Focusing on our specific contributions, we mainly compare scores of the IKA-SST with the FFT IKA-SST, and the SST with the FFT rSVD SST, not the SST with the IKA-SST. With our measurements we aim to demonstrate our algorithmic speed improvements of SST and IKA-SST, while quantifying the accuracy-speed trade-off, aiming for a low scoring error. Nevertheless, our quantitative measurements allow us to clearly visualize the difference in SST and IKA-SST, which might be interesting to practitioners, as it is only qualitatively discussed in the original paper [3]. Fig. 7 clearly visualizes the scoring difference by plotting the approximated scores as distributions over the ground truth scores. It shows how the IKA-SST decouples from the true score for higher values. The other methods mostly follow the optimal line. As previously shown, this behaviour of the IKA-SST is not introduced by our modifications.

TABLE 2: Computational complexity of SST variants with time series length L, window size N, target rank k, number of Lanczos iterations g, oversampling parameter p, and number of subspace iterations q. Typically $g \sim k$ for convergence of the IRLB decomposition.

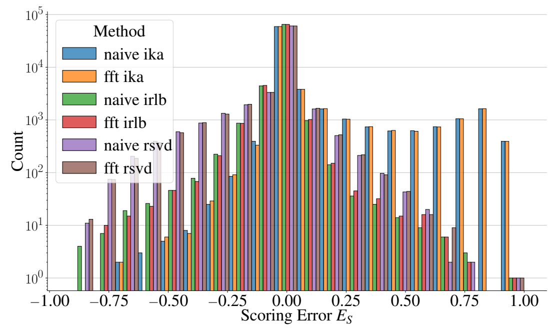
Method	Time Complexity	Corresp. Sec.
Naive SST [2]	$\mathcal{O}\left(LN^3\right)$	II-A
IKA-SST [3]	$\mathcal{O}\left(LN^3\right)$	II-A
SST + rSVD	$\mathcal{O}\left(Lq(k+p)N^2\right)$	III-F
SST + IRLB	$\mathcal{O}\left(LgN^2\right)$	IV-E
SST + rSVD + FFT	$\mathcal{O}\left(Lq(k+p)N\log N\right)$	III-F
SST + IRLB + FFT	$\mathcal{O}\left(LgN\log N\right)$	IV-E
IKA-SST + FFT	$\mathcal{O}\left(LkN\log N\right)$	III-G

TABLE 3: Mean values of BIC, AIC, and Noise Threshold measured using all datasets.

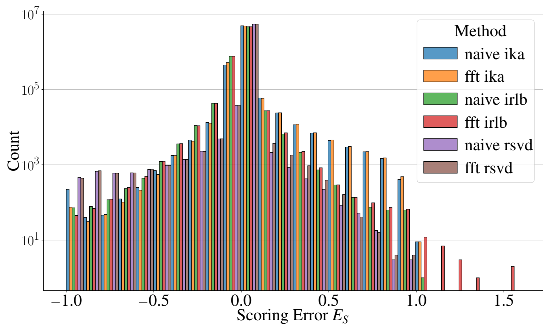
Dataset	BIC [38]	AIC [38]	Noise Threshold [36]
Synthetic Signals	11.86	14.03	16.53
Real Signals	10.56	15.16	17.00

TABLE 4: Mean absolute scoring error of SST variants relative to SVD baseline.

Scoring Error	Synthetic Signals	Real Signals
Naive IKA [3]	71.74×10^{-3}	9.695×10^{-3}
FFT IKA (this paper)	71.63×10^{-3}	9.672×10^{-3}
Naive IRLB [27]	16.83×10^{-3}	8.071×10^{-3}
FFT IRLB [39]	16.89×10^{-3}	8.124×10^{-3}
Naive rSVD [22]	36.20×10^{-3}	1.239×10^{-3}
FFT rSVD (this paper)	35.95×10^{-3}	1.392×10^{-3}



(a) Synthetic Signals.



(b) Real Signals.

FIGURE 6: Scoring error E_S distribution. Computed as the approximated score subtracted from the exact score $S_i - \hat{S}_i$ (see (8)). The error is positive when the approximation underestimates the score ($\hat{S}_i < S_i$) and negative for an overestimation ($\hat{S}_i > S_i$.)

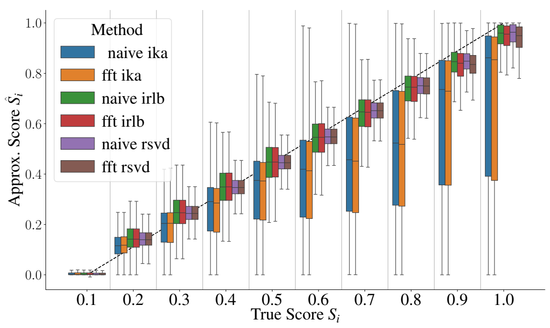


FIGURE 7: Approximated scores over the exact scores for the real signal dataset. The dashed line shows perfect equivalence.

F. RUNTIME SCALING AND PARALLEL EFFICIENCY

We discussed the theoretical complexity improvements for both the SST and IKA-SST in previous sections (Sec. III-F & III-G). Nevertheless, the algorithmic complexities only quantify asymptotic behaviour. We are interested in reducing the SST runtime. Consequently, we have to measure how the asymptotic improvement transfers to the practical application of the algorithms. We evaluate the runtime for our improved versions in comparison to the normal versions of the SST and IKA-SST. Recall that we improved the SST by replacing the complete SVD with our specialized rSVD (FFT rSVD SST, Sec. III-F) and the IKA-SST by never explicitly computing the correlation matrix and employing the fast Hankel matrix product (Sec. III-G).

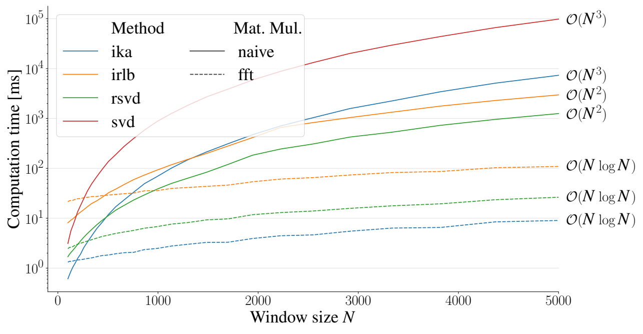
The absolute execution time will vary with the processor, programming language, and other system parameters (Sec. IV-B). Consequently, our measurements are mainly meaningful in relation to each other. For absolute timings, the implementation may have further optimization potential, which is beyond the scope of this paper. The runtime does not depend on time series structure. The execution times were measured on the AMD processor for the same matrix sizes as in the previous Section IV-E. We use the simulation from Sec. IV-A to create the matrices. To quantify the advantages of parallelization, we also vary the amount of threads.

Runtime Scaling. The main results of the runtime measurement are shown in fig. 8. The runtime is measured for one step of the loop in alg. 1. For both plots, the maximum number of threads was fixed to ten.

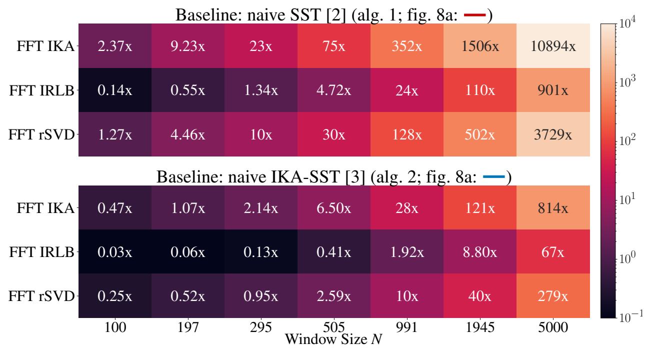
Fig. 8a shows the absolute runtime for different window sizes N and allows us to compare all variants within one plot. The fastest method, our FFT IKA-SST, overtakes all other methods approximately $N = 200$. The FFT rSVD SST is the second fastest method. At a window size of $N \approx 300$, it overtakes even the original IKA-SST. Until these points, the naive computations are faster, mostly because the FFT is a costly operation. The constant FFT cost materializes in the offset at the beginning of each dashed line. Comparing the IRLB with the rSVD, it can be seen that that the rSVD is faster. The plot also shows the algorithmic advantage of employing the fast Hankel matrix product. For example, the specialized FFT rSVD (dashed green) is 300x faster than the naive rSVD (solid green) at $N = 5000$. For large window sizes, the slope of the lines allow to clearly distinguish the algorithmic complexities (logarithmic axis). Due to special, parallel matrix multiplications for smaller matrices, the slopes of lines representing the naive algorithms take some time to settle, while the slopes are nearly constant for the improved SST versions.

Fig. 8b consolidates our runtime improvements by comparing our proposed versions of the SST and IKA-SST with the original algorithms. To compute the runtime factors, we divide the runtime of the original algorithm runtime by that of the improved variant. Improving the IKA-SST with the fast Hankel matrix product (FFT IKA-SST), yields the largest runtime improvement. The reduction of the original $\mathcal{O}(N^3)$ complexity to $\mathcal{O}(N \log N)$, manifests in a speedup of 814x for a window size of 5000. Compared to the naive SST, we see a speedup of more than four order of magnitude (10894x). Compared to the naive SST, both the FFT IKA-SST and the FFT rSVD SST are faster for all window sizes.

Parallelization. Tab. 5 lists the timings for the IRLB, the rSVD and the SVD for a window size of 5000 and different thread numbers. With respect to the parallelization, using ten threads for the rSVD results in 40 % of the execution time



(a) Execution times of the different methods for one SST step. Lower times are better.



(b) Improved- versus naive SST (fig. 8a: ··· vs. —). Numbers > 1x indicate that the method on the left is faster than the baseline.

FIGURE 8: Comparing the scalability of the SST with different algorithmic components (10 threads).

TABLE 5: Execution times of iterative IRLB SST, rSVD SST and naive SVD SST for different thread limits ($N = 5000$).

Threads	**FFT rSVD** (ours)	**FFT IRLB**	**SVD** (seconds!)
1	66.04 ms	112.79 ms	99.56 s
2	42.81 ms	105.81 ms	99.43 s
4	31.50 ms	108.50 ms	98.36 s
6	27.93 ms	107.74 ms	98.05 s
8	25.54 ms	108.73 ms	98.18 s
10	26.34 ms	108.98 ms	98.22 s

TABLE 6: Pairwise execution time comparisons among $\mathcal{O}(N \log N)$ SSTs for different thread limits. A value > 1 indicates that the method in the denominator is faster.

Threads	$\frac{\text{FFT rSVD}}{\text{FFT IKA}}$	$\frac{\text{FFT IRLB}}{\text{FFT IKA}}$	$\frac{\text{FFT IRLB}}{\text{FFT rSVD}}$
1	5.40	15.00	2.99
2	4.02	15.00	3.94
4	3.11	14.95	4.95
6	2.79	15.07	5.54
8	2.63	14.87	5.78
10	2.52	15.05	6.19

compared to single-threaded execution. This parallelizability is a significant advantage of the rSVD compared to other truncated methods, such as the IRLB, especially on modern hardware. The parallelization yields diminishing returns and ceases to decrease after eight threads. Parallelization is only possible over the $k + p$ columns of the subspace matrix. A suitable rule of thumb could be one thread per two columns (we set $k + p = 15$ for our measurements). However, this may be vary with the implementation details (Sec. IV-B).

Comparison of Efficient Variants. Finally, we compare the runtime efficiency of the three $\mathcal{O}(N \log N)$ methods: the FFT rSVD SST, the FFT IKA-SST, and the FFT IRLB SST. For this purpose, we create comparison factors by dividing the execution time of the slower method by the execution time of the faster method and average this value for all window sizes, creating pairwise runtime factors. Tab. 6 presents these runtime factors. Standard deviations were $\leq$ 0.25 for all window sizes and thread limits.

We clearly see that our FFT IKA-SST is the fastest method, but computing the more accurate rSVD SST comes close, when multithreading is allowed. For ten threads, the IKA-SST is only approximately $2.5\times$ faster. Looking at the last column, it is again notable how the rSVD benefits from parallelization when compared with the IRLB.

In summary, the experiments clearly show that the theoretical improvements transfer to the actual implementation. The methods are significantly faster for growing window sizes, while our algorithmic choices do not cause major approximation errors.

G. APPLICATION EXAMPLE

It is beyond the scope of this paper, to evaluate the SST for CPD. Nevertheless, we want to give a quick demonstration of how our improvement impacts the application of the SST. For our use case, we turn to the MIT-BIH Arrhythmia Database [40], which is a common dataset for medical signal processing. It contains the Electrocardiogram (ECG) recordings of several patients. As common in biomedical signals, the shape of the signal is important for the medical assessment, necessitating relatively high sampling rates. The signals are sampled with 360 Hz and are 30 minutes long, containing approximately 650 000 samples per recording.

For brevity, we focus on the IKA-SST and only one patient (patient 234, lead V1), where the signal contains two annotations of changing signal quality ($\sim$) sourced from [41]. To detect these, several heartbeats must be incorporated within the SST detection window. Based on initial tests, we selected 10 heartbeats. In the case of a normal resting heartbeat at 60 bpm, this requires a window size of $N = 360 * 10/2 = 1800$, as the sliding window of the SST is of size $2N - 1$. We run our experiments on a standard Google Colab instance (Intel Xeon Platinum 8259CL CPU @ 2.50GHz) and compute the change score at every fifth sample of the signal.

While the original IKA-SST requires more than 22 hours of computations, our improved version only requires around 22 minutes, running faster than the recording itself. This substantially reduces analysis runtime for applications building on the scoring results and allows online scoring even on commodity hardware.

V. DISCUSSION AND OUTLOOK

A. DISCUSSION

We set out with the initial goal of improving the SST and IKA-SST in terms of speed and scaling with the window size. We rely on the rSVD and the fast Hankel matrix product for these improvements and argued that the rSVD as a truncated decomposition is especially suitable (Sec. III-F & III-H). In our evaluation, we empirically demonstrated the following:

1) Section IV-C: Time series Hankel matrices have a fast decaying singular spectrum.
2) Section IV-D: The rSVD achieves a low approximation error for time series Hankel matrices.
3) Section IV-D: How we chose the parameters of the rSVD and how they affect the approximation error.
4) Section IV-E: In comparison with other competitors, the scoring error of the FFT rSVD SST is fairly small.
5) Section IV-E: The IKA-SST underestimates the score, but this is not due to our algorithmic modifications.
6) Sections IV-D and IV-E: Using the fast matrix product introduces no significant errors.
7) Section IV-F: The asymptotic complexities translate to runtime improvements at fairly small window sizes. The acceleration factors only increase with the window size.
8) Section IV-F: Parallelizing the FFT rSVD SST improves runtime compared to the sequential IRLB.
9) Section IV-F: The FFT IKA-SST is the fastest algorithm (for window sizes ≥ 200, given our implementation).
10) Sections IV-D and IV-F: The FFT rSVD SST out-scales the original IKA-SST, while being more accurate.

Overall, the evaluations support our theoretical arguments and hypotheses. We conclude that the rSVD is a suitable candidate for improving the naive SST. It is faster than other sequential decompositions, such as the IRLB, while maintaining a comparable accuracy and supporting parallel operations. The FFT IKA-SST is the fastest of all methods, but the scoring suffers from the same flaw as the original IKA-SST: it sometimes underestimates the change score. However, this flaw is independent of our improvements. If the SST-based CPD was suitable before, the FFT SST and the FFT IKA-SST are drop-in replacements. For growing window sizes, our modified versions are significantly faster, and only incur a minor approximation error, as expected from the theory and shown in the evaluation.

We recommend the FFT IKA-SST when a low execution time is the highest priority and the FFT rSVD SST, for the best speed-accuracy trade-off. Although the improved IKA-SST is the fastest available method, there are several reasons for using the rSVD SST. It is slightly slower, but the scaling is equal to the IKA-SST, and the scoring is significantly more accurate. Furthermore, the IKA-SST is restricted to computing the change score as the similarity of the largest future singular vector with the singular vectors of the past Hankel matrix. Other methods for computing the change score, such as subspace angles between future and past Hankel matrix, require more than the largest future singular vector, which IKA does not allow. The rSVD SST does not have this restriction and allows other scoring functions, e.g., [14], [15].

The fast matrix product is key in reaching our final asymptotic complexity and introduces no significant numerical errors. However, the involved FFT is expensive and the main reason for the constant offset in the runtime measurements. This offset determines the window size beyond which the proposed algorithms out-scale their original counterparts. The offset is clearly visible in fig. 8a. The dashed lines are initially higher than their solid counterparts. With our limited optimizations, we already reached a reasonably small window size, where the improvements overtake. While we chose Python for reproducibility and its ecosystem, implementing the algorithms in a speed-optimized language closer to the hardware could further reduce this offset. Section V-B lists some ideas to further improve the efficiency independent of the implementation language. Owing to the FFT representation, we never have to construct the complete Hankel matrix, which also saves time. Acquiring memory and filling the Hankel matrix scales with $\mathcal{O}(N^2)$. The linear space complexity was not our initial concern, but is a welcome by-product.

Halko et al. [22] mention that the rSVD with the same computational budget is less accurate than, e.g., IRLB, in most applications. Our empirical results suggest that this difference is almost negligible for time series Hankel matrices. We argued that this is mainly because of their spectral properties as discussed in Section III-H and measured in Section IV-D. In combination with the numerical stability, configurable error bounds, and parallelization, we argue that the advantages of the rSVD outweigh minor differences in accuracy. Nevertheless, the IRLB remains a valid alternative. The rSVD used in this paper is not the only method to compute the truncated SVD using randomized subspaces. There are other, more recent methods using block Krylov approximations and different subspace extractions [42], [43]. We decided on the rSVD as it is well understood in terms of decomposition errors, and the implementation is straightforward.

Although we illustrate our methods using square Hankel matrices $H \in \mathbb{R}^{N\times N}$ for clarity, the same approaches directly extend to rectangular Hankel matrices $H \in \mathbb{R}^{M\times N}$ that arise in many SST-based CPD variants. In this case, the FFT-based Hankel products and randomized decompositions are unchanged; replace N by $\min\{M, N\}$ in the complexity terms and error bounds to account for rectangular matrices.

There are also algorithms that perform SSA-based CPD on multivariate signals. In some cases this results in multilevel Hankel matrices, e.g. [16]. The fast Hankel matrix product can be applied to multilevel Hankel matrices, by performing several independent matrix multiplications for each contained Hankel matrix [19]. Therefore, alg. 5 can also be a adapted to multilevel Hankel matrices. The discussion of the decomposition error for multilevel Hankel matrices are beyond the scope of this paper, but our results indicate that the Hankel matrix product does not introduce large, numerical errors into the rSVD, which should remain the case for multilevel Hankel matrices.

Our improvements address the computational bottleneck of SST and IKA-SST. However, they do not alter the inherent domain of applicability of these methods. SST-based CPD is particularly effective for structured signals that exhibit repeating patterns, seasonality, or quasi-stationary regimes, where Hankel matrices display a rapidly decaying singular spectrum [6]. In contrast, stochastic signals or signals with chaotic dynamics often lead to slowly decaying spectra, reducing the effectiveness of low-rank approximations and degrading the quality of SST-based change scores. Partly this can be offset by increasing p and q, but both induce additional computational effort. In such cases, alternative methods may be preferable. Examples include relative density estimation [7], Bayesian Online CPD [44], Gaussian process CPD [45], and parametric methods [46]. Our contributions should be understood as enabling scalable SST-based CPD within its natural regime, rather than serving as a universal CPD solution. In practice, the choice between SST and alternative detectors should be guided by the signal structure and detection philosophy.

Nevertheless, the SST and IKA-SST are widely applied algorithms in different scenarios. Our contribution ensures that both are faster for growing window sizes and can now be applied in high-frequency or long-horizon scenarios where they were previously computationally prohibitive.

B. OUTLOOK

Our analysis of the FFT rSVD concentrates mainly on its applicability for time series CPD. As mentioned in Section II, there are further potential applications for the FFT rSVD in system identification or dynamical systems analysis. We also provide some thoughts on a cheap online estimation of errors in the case of positive semi-definite Hankel matrices that are outside of the scope of our paper (Sec. III-H). Detailed parametrization of the method, i.e., q and p for the rSVD, as well as numerical error introduced by the FFT, need to be evaluated for each application separately. As noted in Section III-F, the FFT rSVD extends to any matrix that is a product of Hankel matrices when the generating Hankel matrices are known. Ye et al. [47] prove that any matrix can be constructed as a product of Hankel matrices. In the case of a small number of Hankel factors, the fast Hankel matrix product may speed up the computation. Unfortunately, there is no known method to efficiently construct these Hankel factors [48].

There is literature on how to improve the fast Hankel matrix product [49]. Potentially, their ideas could decrease the overhead of the FFT as the main step in the fast matrix product, further decreasing the threshold after which the improved algorithms out-scale the original versions. For the complete scoring of a time series, the SST repeatedly computes separate decompositions on largely overlapping signal parts. This may leave room for further improvements. For example, sliding-window FFTs could potentially further decrease the runtime, e.g., using the algorithm in [50]. This idea of a sliding-window FFT directly interconnects with the application of the SST to streamed time series. Naively, the SST can be used on streaming time series by collecting the values in a rolling buffer of size $2N-1$, repeating lines 3-7 of alg. 1 for newly arriving samples. With this setup, it is crucial that the algorithms run faster than newly arriving samples. Our application example (Sec. IV-G) shows that our improvements enable buffered processing even for larger window sizes. With a sliding-window FFT, the runtime may decrease further, because the FFT would not process the complete window repeatedly. We consider these aspects for future work.

VI. CONCLUSION

Due to the involvement of SVD, the per-step runtime of the SST [2] cubically grows $\left(\mathcal{O}(N^3)\right)$ with the window size N. While the IKA-SST [3] eliminates the SVD bottleneck, the cubic complexity remains. Consequently, SST-based applications are limited to small window sizes or forced to implement extensive preprocessing steps, adding further hyperparameters to the pipeline. For long-duration patterns or high-frequency signals, applications often require larger window sizes to span the entire region of interest.

We tackle this scaling problem by utilizing the rSVD [22] and a fast Hankel matrix product to reduce the computational complexity of both methods to $\mathcal{O}(N \log N)$, and the space complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Except for minor approximation errors, our algorithmic modifications do not alter the detection outcome, but they significantly reduce the algorithmic runtime as the window sizes increases.

To substantiate our algorithmic choices, we provide theoretical arguments for the suitability of the rSVD for SST-based CPD. By discussing the fast spectral decay of Hankel matrices and linking the decomposition error with the scoring error via a novel error bound, we demonstrate why the more efficient rSVD incurs only a small approximation error, when replacing the SVD in the SST.

Empirical measurements on a large body of real and synthetic time series confirm our claims: the expected properties hold, score error remains small, and the asymptotic gains translate into wall-time improvements at moderate N. Our modified variants out-scale the naive SST, and even the IKA-SST. We measure runtime improvements of up to two and three orders of magnitude for IKA-SST (814x) and SST (3729x), respectively (fig. 8). Applying the accelerated IKA-SST to detect quality changes within an ECG signal demonstrates that our proposed modifications reduce detection time from several hours to mere minutes (22 h $\rightarrow$ 22 min, Sec. IV-G). We release the code for our experiments and provide reference implementations to facilitate further applications.

Among the accelerated variants, the FFT IKA-SST is the fastest method, while the FFT rSVD SST offers the best trade-off in terms of speed and accuracy (Sec. V).

In summary, our algorithmic modifications substantially improve the scalability of the SST and IKA-SST. They enable significantly larger window sizes within the same computational budget, reducing the runtime of current applications, and extending the applicability of the SST-based CPD to high-frequency or long-horizon scenarios.

REFERENCES

[1] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, pp. 339–367, 5 2017.

[2] T. Idé and K. Inoue, "Knowledge discovery from heterogeneous dynamic systems using change-point correlations," in *Proceedings of the SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 5 2005, pp. 571–575.

[3] T. Idé and K. Tsuda, "Change-point detection using krylov subspace learning," in *Proceedings of the SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 5 2007, pp. 515–520.

[4] L. Weber and R. Lenz, "Machine learning in sensor identification for industrial systems," *it - Information Technology*, vol. 65, pp. 177–188, 2023.

[5] S. Wu *et al.*, "Online adaptive anomaly detection in networked electrical machines by adaptive enveloped singular spectrum transformation," *IEEE Internet of Things Journal*, 2024.

[6] V. Moskvina and A. Zhigljavsky, "An algorithm based on singular spectrum analysis for change-point detection," *Communications in Statistics - Simulation and Computation*, vol. 32, pp. 319–352, 5 2003.

[7] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 5 2013.

[8] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh, "Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels," in *IEEE International Conference on Data Mining (ICDM)*, vol. 2017-November. IEEE, 5 2017, pp. 117–126.

[9] T. D. Ryck, M. D. Vos, and A. Bertrand, "Change point detection in time series data using autoencoders with a time-invariant representation," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3513–3524, 5 2021.

[10] N. Keriven, D. Garreau, and I. Poli, "Newma: A new method for scalable model-free online change-point detection," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3515–3528, 2020.

[11] A. Ermshaus, P. Schäfer, and U. Leser, "Clasp: parameter-free time series segmentation," *Data Mining and Knowledge Discovery*, vol. 37, pp. 1262–1300, 5 2023.

[12] G. Romano, I. A. Eckley, and P. Fearnhead, "A log-linear nonparametric online changepoint detection algorithm based on functional pruning," *IEEE Transactions on Signal Processing*, vol. 72, pp. 594–606, 2024.

[13] J. Yu, Y. He, Q. Yan, and X. Kang, "Specview: Malware spectrum visualization framework with singular spectrum transformation," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5093–5107, 2021.

[14] Y. Mohammad and T. Nishida, "Robust singular spectrum transform," in *International Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems*. Springer, 2009, pp. 123–132.

[15] Y. Mohammad and T. Nishida, "On comparing ssa-based change point discovery algorithms," in *2011 IEEE/SICE International Symposium on System Integration*. IEEE, 2011, pp. 938–945.

[16] A. Alanqary, A. Alomar, and D. Shah, "Change point detection via multivariate singular spectrum analysis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 218–23 230, 2021.

[17] A. Korobeynikov, "Computation- and space-efficient implementation of ssa," *Statistics and Its Interface*, vol. 3, pp. 357–368, 2010.

[18] P. C. Rodrigues, P. G. S. E. Tuy, and R. Mahmoudvand, "Randomized singular spectrum analysis for long time series," *Journal of Statistical Computation and Simulation*, vol. 88, pp. 1921–1935, 5 2018.

[19] S. Sahnoun, K. Usevich, and P. Comon, "Multidimensional esprit for damped and undamped signals: Algorithm, computations, and perturbation analysis," *IEEE Transactions on Signal Processing*, vol. 65, pp. 5897–5910, 5 2017.

[20] Y. Yang and J. Rao, "Robust and efficient harmonics denoising in large dataset based on random svd and soft thresholding," *IEEE Access*, vol. 7, pp. 77 607–77 617, 2019.

[21] H. Wang and J. Anderson, "Large-scale system identification using a randomized svd," in *American Control Conference (ACC)*. IEEE, 5 2022, pp. 2178–2185.

[22] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, pp. 217–288, 5 2011.

[23] T. K. Boehme and R. Bracewell, "The fourier transform and its applications." *The American Mathematical Monthly*, vol. 73, p. 685, 5 1966.

[24] T. G. Stockham Jr, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, 1966, pp. 229–233.

[25] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert, "Algorithm 971: An implementation of a randomized algorithm for principal component analysis," *ACM Transactions on Mathematical Software*, vol. 43, pp. 1–14, 5 2017.

[26] J. Dongarra *et al.*, "The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale," *SIAM review*, vol. 60, no. 4, pp. 808–865, 2018.

[27] J. Baglama and L. Reichel, "Augmented implicitly restarted lanczos bidiagonalization methods," *SIAM Journal on Scientific Computing*, vol. 27, pp. 19–42, 5 2005.

[28] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 5 1936.

[29] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The Quarterly Journal of Mathematics*, vol. 11, pp. 50–59, 1960.

[30] J. Gillard and K. Usevich, "Hankel low-rank approximation and completion in time series analysis and forecasting: a brief review," *Statistics and Its Interface*, vol. 16, pp. 287–303, 2023.

[31] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*. London: Academic Press, 1990.

[32] P.-Å. Wedin, "Perturbation bounds in connection with singular value decomposition," *BIT Numerical Mathematics*, vol. 12, no. 1, pp. 99–111, 1972.

[33] H. A. Dau *et al.*, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, pp. 1293–1305, 5 2019.

[34] C. R. Harris *et al.*, "Array programming with numpy," *Nature*, vol. 585, pp. 357–362, 5 2020.

[35] S. K. Lam, A. Pitrou, and S. Seibert, "Numba," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. ACM, 5 2015, pp. 1–6.

[36] M. Gavish and D. L. Donoho, "The optimal hard threshold for singular values is $4/\sqrt{3}$," *IEEE Transactions on Information Theory*, vol. 60, pp. 5040–5053, 5 2014.

[37] B. W. Lewis and M. Kane, "Irlbpy," https://github.com/bwlewis/irlbpy, 2018, accessed: 03.06.2024.

[38] N. Golyandina, "Particularities and commonalities of singular spectrum analysis as a method of time series analysis and signal processing," *WIREs Computational Statistics*, vol. 12, no. 4, p. e1487, 2020.

[39] K. Browne, S. Qiao, and Y. Wei, "A lanczos bidiagonalization algorithm for hankel matrices," *Linear Algebra and its Applications*, vol. 430, pp. 1531–1543, 5 2009.

[40] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[41] A. Roman, H.-M. D. of Health Sciences, Technology, R. G. Mark, and G. B. Moody, "Mit-bih arrhythmia database," https://www.kaggle.com/datasets/protobioengineering/mit-bih-arrhythmia-database-modern-2023, 2025, accessed: 08.08.2025. [Online]. Available: https://www.kaggle.com/dsv/12526755

[42] C. Musco and C. Musco, "Randomized block krylov methods for stronger and faster approximate singular value decomposition," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 1, 5 2015, pp. 1396–1404.

[43] M. F. Kaloorazi and R. C. de Lamare, "Subspace-orbit randomized decomposition for low-rank matrix approximations," *IEEE Transactions on Signal Processing*, vol. 66, pp. 4409–4424, 5 2018.

[44] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[45] Y. Saatçi, R. D. Turner, and C. E. Rasmussen, "Gaussian process change point models," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 927–934.

[46] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[47] K. Ye and L.-H. Lim, "Every matrix is a product of toeplitz matrices," *Foundations of Computational Mathematics*, vol. 16, pp. 577–598, 5 2016.

[48] K. Ye, "New classes of matrix decompositions," *Linear Algebra and Its Applications*, vol. 514, pp. 47–81, 5 2016.

[49] K. Ye and L.-H. Lim, "Algorithms for structured matrix-vector product of optimal bilinear complexity," in *IEEE Information Theory Workshop (ITW)*. IEEE, 9 2016, pp. 310–314.

[50] Z. Rafii, "Sliding discrete fourier transform with kernel windowing," *IEEE Signal Processing Magazine*, vol. 35, pp. 88–92, 5 2018.

[51] S. O'Rourke, V. Vu, and K. Wang, "Random perturbation of low rank matrices: Improving classical bounds," *Linear Algebra and its Applications*, vol. 540, pp. 26–59, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0024379517306420

[52] A. K. Saibaba, "Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms," *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 1, pp. 23–48, 2019.

[53] B. Beckermann and A. Townsend, "On the singular values of matrices with displacement structure," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, pp. 1227–1248, 5 2017.

REPRODUCIBILITY

The code and data used to create this paper are publicly available (https://github.com/Lucew/approximate_hankel). The reference implementations are available through the *changepoynt* package hosted on PyPi (https://pypi.org/project/changepoynt/). You can install them using the command *pip install changepoynt*.

ACKNOWLEDGMENT

The authors would like to thank Prof. Eamonn Keogh and everyone who contributed to the UCR time series archive, used in this paper. We also thank the editor and the reviewers for their valuable comments and suggestions. SIEMENS Energy AG supported this work in the context of Project DARTS. We thank Martin Buehl and Markus Zenker for their valuable insights into industrial data analytics.

LUCAS WEBER studied Electrical Engineering and Information Technology at the Technical University of Dresden. He is currently a research assistant at the chair of Evolutionary Data Management in the Computer Science Department of the Friedrich-Alexander Universität Erlangen-Nürnberg. His research interests include pattern recognition and signal processing for robust machine learning systems as well as knowledge discovery and mining in time series.

RICHARD LENZ is Professor for Evolutionary Data Management in the Department of Computer Science at the Friedrich-Alexander Universität Erlangen-Nürnberg. He leads research on evolutionary information systems, data quality and integration, workflow management, and research data management. He received his Ph.D. from the Friedrich-Alexander Universität Erlangen-Nürnberg with a thesis on "Adaptive Data Replication in Distributed Systems" and his habilitation at the University of Marburg for his work on "Evolutionary Information Systems in Healthcare".

APPENDIX. A

We use dedicated notation to separate special variables throughout the paper. For better readability, we list the most important symbols that are not standard notation in the following table. We introduce all symbols in the paper itself. It is self-contained without this overview.

Special Notations	Explanation
SST & IKA-SST	
Matrix H	Hankel matrix introduced in Section III-A.
Matrix C	Correlation matrix $C = HH^T$.
Matrix T	Time Series.
Matrix S	Score Series (the same size as T).
Vectors $\alpha, \beta, \mathbf{q}, \mathbf{r}$	Intermediate vectors of the IKA-SST (alg. 2).
Scalar N	Window size of the SST. Equal to the window size for the SVD matrices (see Sec. III-C).
Scalar k	Truncation rank, a main hyperparameter of the SST. It also occurs in the IKA-SST (alg. 2) and the rSVD (algs. 3 & 5), as they are used in the SST.
Scalar L	Time series length.
Scalar l	Sliding-window size.
Scalar S_i	The score of a single SST step (alg. 1, line 7).
Scalar t	A point in time.
Scalar δ	Time gap between the two Hankel matrices (alg. 1).
Scalar τ	Time gaps between sliding windows.
Scalar ζ	Lanczos rank in the IKA-SST (alg. 2).
SVD & rSVD	
Matrix U	Matrix containing the left singular vectors.
Matrix Σ	Matrix containing the singular values in decreasing order on the main diagonal.
Matrix V	Matrix containing the right singular vectors.
Matrix Q	The subspace projector of the rSVD (Sec. III-D).
Matrix Ω	The sampling matrix of the rSVD (Sec. III-D).
Vector $\mathbf{u}$	A left singular vector.
Scalar σ	A singular value. Singular values are always sorted in decreasing order ($\sigma_i \geq \sigma_j \; \forall i \leq j$).
Scalar k	Truncation rank. Equivalent to k from the SST section, as the rSVD is used as part of the SST.
Scalar p	Oversampling parameter of the rSVD (Sec. III-D).
Scalar q	Number of rSVD subspace iterations (Sec. III-D).
Errors	
Symbol E	General character to denote an error term.
Scalar E_S	Scoring Error (see (8)).
Scalar E_D	Decomposition Error (see (13)).
Scalar E_f	Decomposition Error of the future Hankel matrix H_f.
Scalar E_p	Decomposition Error of the past Hankel matrix H_p.
Scalar $\mathbb{E}^*_{\text{Proj}}$	Decomposition Error of the rSVD (see (4), notation taken from Halko et al. [22, Corollary 10.10 f]).
Sub- & Superscripts	
Subscript $\square_p$	Denotes variables that are linked to the past Hankel matrix H_p (alg. 1).
Subscript $\square_f$	Denotes variables that are linked to the future Hankel matrix H_f (alg. 1).
Subscript $\square_k$	Denotes truncated results of the SVD or rSVD.
Superscript $\hat{\square}$	Denotes approximations.
Superscript $\tilde{\square}$	Denotes intermediate results.
Miscellaneous	
Matrix R	Result of the fast Hankel matrix product (alg. 4).
Matrices A, X	Arbitrary matrices with no special structure.
Matrices B, Y	Intermediate results in the rSVD (alg. 3).
Vector $\mathbf{a}$	Arbitrary vector with no special structure.
Scalar G	Arbitrary matrix size in alg. 4
Symbol $\mathscr{F}(\square)$	Fast Fourier Transform (FFT).
Symbol $[a : b]$	Extract values a to b from a vector or matrix of size $1 \times \square$.

APPENDIX. B

We now provide the proof of (9) in Section III-I. The equation ties the decomposition error to the final scoring error of the SST, providing a clean connection between the employed decomposition and the final scoring error.

Notations and general statements for the following equations:

- $H_p,\ H_f \in \mathbb{R}^{N\times N}$ are the past/future Hankel matrices
- $U_p \in \mathbb{R}^{N\times k}$ contains the top-k orthonormal left singular vectors of H_p
- $\mathbf{u}_f \in \mathbb{R}^{N\times 1}$ is the top left singular vector of H_f
- $\sigma_{p,i}/\sigma_{f,i}$ are the singular values of H_p/H_f sorted in descending order ($\sigma_{\cdot,i} > \sigma_{\cdot,i+1} \forall i$)
- $\hat{U}_p,\ \hat{\mathbf{u}}_f$ are approximation of the original singular vectors (e.g., by using the FFT rSVD)
- Score $\hat{S}_i \rightarrow f(\hat{U}_p,\ \hat{\mathbf{u}}_f)$ is the approximation of Score $S_i \rightarrow f(U_p, \mathbf{u}_f)$
- All norms $||\bullet||$ in the following equation will be 2-norms

We start at the absolute scoring error $|E_S|$ that compares the two SST scores:

$$\begin{aligned}|E_S| :=& |S_i - \hat{S}_i| = |(1-||U_p^T\mathbf{u}_f||^2) - (1-||\hat{U}_p^T\hat{\mathbf{u}}_f||^2)|\\ =& |(||U_p^T\mathbf{u}_f||^2 - ||\hat{U}_p^T\hat{\mathbf{u}}_f||^2)|\end{aligned} \tag{14}$$

We will adapt (14) until we can use Wedin's general sin-theorem [32] from matrix perturbation theory. The theorem bounds the largest canonical angle between subspaces of the original and perturbed matrices. In our case, we work with the column subspace of H_p/H_f. The left singular vectors $U_p/\mathbf{u}_f$ are an orthonormal basis for this subspace.

As a first step, we want to replace the orthonormal bases U_p^T and $\mathbf{u}_f$ with their corresponding subspace projectors $P_p = U_pU_p^T$ and $P_f = \mathbf{u}_f\mathbf{u}_f^T \in \mathbb{R}^{NxN}$. Before doing so, we prove that this replacement does not change both norms in (14):

$$\begin{aligned}||P_pP_f||^2 &:= ||\underbrace{U_pU_p^T\mathbf{u}_f}_{=:\mathbf{a}}\ \underbrace{\mathbf{u}_f^T}_{=:\mathbf{b}^T}||^2 = ||\mathbf{a}\mathbf{b}^T||^2\\ &= \lambda_{\max}\left((\mathbf{a}\mathbf{b}^T)^T(\mathbf{a}\mathbf{b}^T)\right) = \lambda_{\max}(\mathbf{b}\underbrace{\mathbf{a}^T\mathbf{a}}_{\text{scalar}}\mathbf{b}^T)\\ &= \mathbf{a}^T\mathbf{a}\cdot\underbrace{\lambda_{\max}(\mathbf{b}\mathbf{b}^T)}_{\star} = ||\mathbf{a}||^2\cdot\underbrace{||\mathbf{b}||^2}_{||\mathbf{u}_f||^2=1} = ||\mathbf{a}||^2\\ &= ||P_p\mathbf{u}_f||^2 = ||U_pU_p^T\mathbf{u}_f||^2 = \mathbf{u}_f^T(U_pU_p^T)^TU_pU_p^T\mathbf{u}_f\\ &= \mathbf{u}_f^TU_p\underbrace{U_p^TU_p}_{=I}U_p^T\mathbf{u}_f = \mathbf{u}_f^TU_pU_p^T\mathbf{u}_f\\ &= ||U_p^T\mathbf{u}_f||^2 \text{ q.e.d.}\end{aligned} \tag{15}$$

$\star$: $\mathbf{b}\mathbf{b}^T$ is rank one and has eigenvalue $||\mathbf{b}||^2$

For understanding the proof, note that $||A||^2$ is equal to the largest eigenvalue $\lambda_{\max}$ of (A^TA) [31, p. 69] and $\lambda(cA) = c\lambda(A)$ if c is scalar. Consequently, we can replace U and $\mathbf{u}$ with their corresponding projectors in (8), without changing the result:

$$|E_S| = |(||P_pP_f||^2 - ||\hat{P}_p\hat{P}_f||^2)| \tag{16}$$

We want to apply the reverse triangle inequality ($|(|a|-|b|)| \leq |a-b|$), but have to deal with the squares $||\cdot||^2$ first. For that we replace $a := ||P_pP_f||$ (positive, scalar) and $b := ||\hat{P}_p\hat{P}_f||$ (positive, scalar).

$$|E_S| = |a^2-b^2| = |(a-b)\underbrace{(a+b)}_{\geq 0}| = |a-b|(a+b) \tag{17}$$

We can then use the sub-multiplicativity of the Hölder norms ($||AB|| \leq ||A||\cdot||B||$) [31, p. 69] (here $p = 2$) and the fact that all orthogonal projectors have norm of one $||\hat{P}_p|| = ||P_p|| = ||P_f|| = ||\hat{P}_f|| = 1$, since the matrices $\hat{P}_p$ and P_f are idempotent (largest eigenvalue is one) [31, p. 9]:

$$\begin{aligned}&a = ||P_pP_f|| \leq ||P_p||\cdot||P_f|| = 1\\ &b = ||\hat{P}_p\hat{P}_f|| \leq ||\hat{P}_p||\cdot||\hat{P}_f|| = 1\\ &\text{since } 0 \leq a,b \leq 1 \rightarrow (a+b) \leq 2\end{aligned} \tag{18}$$

Using (18) in (17) and replacing a and b again now yields:

$$|E_S| \leq 2|(||P_pP_f|| - ||\hat{P}_p\hat{P}_f||)| \tag{19}$$

Apply the reverse triangle inequality ($|(|a|-|b|)| \leq |a-b|$) to get both terms in the same norm:

$$|E_S| \leq 2||P_pP_f - \hat{P}_p\hat{P}_f|| \tag{20}$$

Now add and subtract $\hat{P}_pP_f$:

$$\begin{aligned}|E_S| &\leq 2||P_pP_f - \hat{P}_pP_f + \hat{P}_pP_f - \hat{P}_p\hat{P}_f||\\ &= 2||(P_p-\hat{P}_p)P_f + \hat{P}_p(P_f-\hat{P}_f)||\end{aligned} \tag{21}$$

Apply the triangle inequality ($||A+B|| \leq ||A|| + ||B||$, for arbitrary matrices A, B) to separate both terms again:

$$|E_S| \leq 2||(P_p-\hat{P}_p)P_f|| + 2||\hat{P}_p(P_f-\hat{P}_f)|| \tag{22}$$

Again, use the sub-multiplicativity of the norm to eliminate the factor in each of the two norms:

$$|E_S| \leq 2||P_p-\hat{P}_p||\cdot||P_f|| + 2||\hat{P}_p||\cdot||P_f-\hat{P}_f|| \tag{23}$$

with $||\hat{P}_p|| = ||P_f|| = 1$ as explained before:

$$|E_S| \leq 2\underbrace{||P_p-\hat{P}_p||}_{T1} + 2\underbrace{||P_f-\hat{P}_f||}_{T2} \tag{24}$$

Both parts $T1$ and $T2$ are norms of the differences between subspace projectors. Stewart and Sun state that the normed difference is equal to the largest canonical angle between the two subspaces [31, pp. 92&93]:

$$||P-\hat{P}|| = sin\theta_1 = ||sin\Theta|| \tag{25}$$

With this form, we can apply Wedin's Theorem [32, p. 102, (3.1)] which gives us a bound on the canonical angles. In the original equation, one can replace ϵ with the perturbation $||E||$ as noted by Wedin [32, p. 107 below (4.4)]. Since our $||E||$ is random, we use the modified Wedin's theorem of O'Rourke et al. [51]:

$$||P-\hat{P}|| = sin\theta_1 = ||sin\Theta|| \leq 2\frac{||E||}{\Delta} = 2\frac{||E||}{\sigma_k - \sigma_{k+1}} \tag{26}$$

Replacing both $T1$ and $T2$ in (24) with (26), we finally arrive at an upper bound for our scoring error that only depends on the perturbation $||E||$ and the singular values of the original Hankel matrices H_p and H_f (III-I, see (9)):

$$|E_S| \leq \underbrace{4\frac{||E_p||}{\sigma_{p,k} - \sigma_{p,k+1}}}_{2\cdot T1} + \underbrace{4\frac{||E_f||}{\sigma_{f,1} - \sigma_{f,2}}}_{2\cdot T2}$$

Note that the factor 2 could be eliminated, if we would use $||U_p^T \mathbf{u}_f||$ without the square to calculate the score.
The rSVD subspace is not literally the top-k subspace of some $A+E$ matrix, but the projection residual $(I-QQ^T)A$ acts as an effective perturbation; treating it this way, as in Saibaba [52], justifies applying Wedin's theorem to bound the canonical angles. This also extends to other subspace decompositions.

APPENDIX. C

We now derive (12) in Section III-I. The result of the previous section is independent of the decomposition method, but we are using the rSVD with a Gaussian random projector. Consequently, we can also employ bounds specialized for randomized decompositions. Saibaba [52] proposes a bound for $sin\theta_i$. Some of their results are dedicated to comparing truncated subspaces from the exact SVD and the rSVD ($sin\theta_j'$ in the paper):

$$sin\theta_1 \leq \gamma_1 \frac{\gamma_k^{2q}}{1-\gamma_k} ||\Omega_2 \Omega_1^\dagger||^2 \tag{27}$$

$$\text{with } \gamma_i := \frac{\sigma_{k+1}}{\sigma_i} \tag{28}$$

Using the inequality $||\Omega_2\Omega_1^\dagger||^2 \leq C_e$ in the proof of theorem 6 for Gaussian random projectors [52, p. 39], we arrive at the following:

$$sin\theta_1 \leq \gamma_1 \frac{\gamma_k^{2q}}{1-\gamma_k} \left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p} \right) \tag{29}$$

Replace the singular value gaps γ_i:

$$sin\theta_1 \leq \frac{\sigma_{k+1}}{\sigma_1} \frac{\left(\frac{\sigma_{k+1}}{\sigma_k}\right)^{2q}}{1-\frac{\sigma_{k+1}}{\sigma_k}} \left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p} \right) \tag{30}$$

By replacing $T1$ and $T2$ in (24) with the previous equation, we arrive at the final bound for the scoring error (III-I, see (12)):

$$\begin{aligned}|E_S| \leq & 2\frac{\sigma_{p,k+1}}{\sigma_{p,1}} \frac{\left(\frac{\sigma_{p,k+1}}{\sigma_{p,k}}\right)^{2q}}{1-\frac{\sigma_{p,k+1}}{\sigma_{p,k}}} \left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p} \right) \\ & + 2\frac{\sigma_{f,2}}{\sigma_{f,1}} \frac{\left(\frac{\sigma_{f,2}}{\sigma_{f,1}}\right)^{2q}}{1-\frac{\sigma_{f,2}}{\sigma_{f,1}}} \left(\frac{1}{\sqrt{p-1}} + \frac{e\sqrt{(1+p)(N-1)}}{p} \right)\end{aligned}$$

APPENDIX. D

We promised additional thoughts on real and positive semi-definite Hankel matrices. As discussed in the following, they are not guaranteed on our case, but other applications might benefit from our insights.
Beckermann et al. [53] prove that all real and positive semi-definite Hankel matrices have rapidly decaying singular values. Their result is an impressive argument as to why low-rank approximations often work so well for different engineering problems. They even derive an analytical expression for the exponential decay of the eigenvalue spectrum:

$$\frac{\sigma_{i+2l}}{\sigma_i} \leq \underbrace{16\left[\exp\left(\frac{\pi^2}{4\log(8\lfloor N/2 \rfloor/\pi)}\right)\right]^{-2l+2}}_{=:f_{dec}^l} \tag{31}$$

This fast decay directly shows that rSVD is especially suitable for positive semi-definite Hankel matrices. The analytical expression even allows to compute an extremely cheap estimate of the approximation error. Halko et al. [22] discuss online error estimation in chapter 4.4 of their paper. Their error estimation comes at the cost of several iterative matrix vector products of the original matrix A with a batch of random vectors. Given positive semi-definite Hankel matrices, the results of Beckermann et al. [53] allow for an even cheaper estimation $\mathbb{E}_k^*$. Combining (6) with (31), we can derive a new upper bounded estimation that comes only at the cost of multiplying two constant factors by the largest eigenvalue σ_0. During computation, we could estimate the largest singular value $\tilde{\sigma}_0$ for an approximation of the bound:

$$\tilde{\mathbb{E}}_{\text{Proj}}^* \lesssim \underbrace{f_{proj}^k}_{\to(4)} * \underbrace{f_{dec}^{k/2}}_{\to(31)} * \tilde{\sigma}_0 \tag{32}$$

This bound can be used to assess the decomposition quality during runtime. This allows the creation of adaptive algorithms that change their parameters depending on the data properties, allowing to specify an expected approximation error instead of the rSVD parameters. It is especially useful because the largest singular vector can be obtained independently, efficiently and accurately.
Unfortunately, not all possible Hankel matrices from time series are positive semi-definite. To show that a Hankel matrix is positive semi-definite, it is sufficient to show that all eigenvalues are positive (the matrix is real, symmetric; therefore, Hermitian). An example of a Hankel matrix with negative eigenvalues can be constructed from a perfect rectangular impulse of amplitude one in an otherwise zero signal:

$$M_{\mathbf{neg}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \text{ Eigenvalues: 1, 1, -1}$$

The example shows that positive semi-definiteness does not generally hold for Hankel matrices. Therefore, focusing on SST-based Hankel matrices in the paper, we study the spectral decay of Hankel matrices empirically.

IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Accelerating Singular Spectrum Transformation for Scalable Change Point Detection

LUCAS WEBER[1], **RICHARD LENZ**[1]
[1]Department of Computer Science, Friedrich-Alexander Universität Erlangen-Nürnberg, 91058 Erlangen, Germany
Corresponding author: Lucas Weber (e-mail: lucas.weber@fau.de).

SIEMENS Energy AG has supported this work in the context of Project DARTS.

ABSTRACT The Singular Spectrum Transformation (SST) is a powerful technique for Change Point Detection (CPD) and is widely applied in time series analysis to identify abrupt structural changes. Due to the involvement of the Singular Value Decomposition (SVD), the runtime of the SST cubically grows ($\mathcal{O}(N^3)$) with the window size N. Therefore, applications building on the SST are often limited to small window sizes. An improved version of the SST, the IKA-SST, removes this bottleneck, but still suffers from similar scaling problems. Utilizing randomized decompositions and a fast matrix product, we reduce the computational complexities of both the SST and the IKA-SST to log-linear ($\mathcal{O}(N \log N)$), enabling efficient CPD for large window sizes and high-frequency signals.
Focusing on the approximation error, we discuss how randomized decompositions are well-suited to SST-based CPD. By linking the scoring error to the decomposition error via a novel error bound, we provide theoretically grounded arguments that the efficiency gains incur only a small approximation error. We empirically evaluate the improved SST algorithms on a diverse collection of real and synthetic time series. Comparisons with the original algorithms confirm that the speedup, parallelization, and low approximation error translate well from theory to practice. Depending on the application, we measure acceleration factors of up to three orders of magnitude, reducing the wall time from hours to minutes, or even seconds, broadening the applicability of SST-based CPD. To support reproducibility and further applications, we publish our experimental code and reference implementations.

INDEX TERMS Change Point Detection (CPD), Signal Processing, Singular Spectrum Analysis (SSA), Hankel Matrix Decomposition, Randomized Singular Value Decomposition (rSVD).

I. INTRODUCTION

Time series analysis is paramount in data science and analytics because it reveals patterns and trends within chronological data and drives informed decision-making processes across various industries. Change Point Detection (CPD), as a research direction in time series analysis, provides methods to find abrupt structural changes in time series, which can be used to detect state changes in the measured systems. CPD finds application in a wide range of real-world problems such as medical condition monitoring, climate change analysis, speech recognition, and human activity analysis [1].
The Singular Spectrum Transformation (SST) [2], [3] is a common CPD algorithm that detects changes in time series shape or patterns. Example applications of the SST include power plants with thousands of time series [4] and high-frequency measurements from electrical machines [5]. The SST estimates a change score by comparing the similarity of characteristic time series components. It derives these components by performing the Singular Value Decomposition (SVD) on a Hankel matrix, constructed from the original time series, and parametrized by a window size (see fig. 1). The window size N is the main hyperparameter of the SST and directly determines the matrix size for the SVD. The computational complexity and, therefore, the runtime of the SVD scales cubically with the matrix size ($\mathcal{O}(N^3)$), increasing rapidly for larger window sizes. The IKA-SST [3] is an improved variant of the SST that removes SVD as the computational bottleneck but requires the computation of a correlation matrix. Similar to the SVD, this step scales poorly with the window size. See Section III-C for further details on both algorithms. Addressing these bottlenecks is the focus of our contribution.

The Problem. In the case of larger patterns or high-frequency signals, the window size increases and the SST runtime quickly becomes infeasible. In practice, this substantially

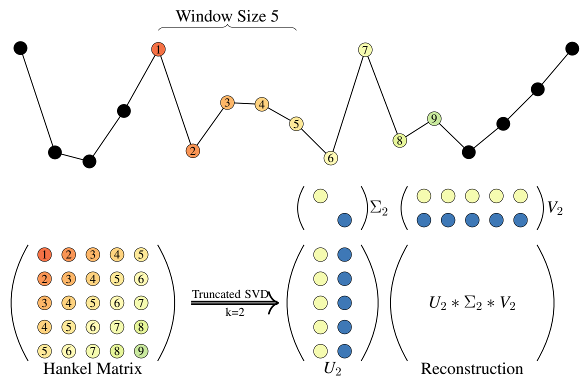


FIGURE 1: Visualization of Hankel matrix construction from a random time series (window size = 5) and truncated SVD reconstruction ($k = 2$).

raises the resource requirements, entails elaborate preprocessing steps, or limits the practical applicability of the SST to smaller patterns or fewer signals.

The Goal. Speed up SST-based change point scoring for increasing window sizes. Specifically, we aim to reduce the runtime of both the SST and IKA-SST to speed up existing applications and broaden their applicability to high-frequency signals and long-duration patterns. We seek to maintain algorithmic structure while reducing computational effort.

To achieve this goal, we employ approximative methods from linear algebra and exploit the special Hankel structure of matrices occurring in both the SST and IKA-SST. In particular, we propose the application of the randomized Singular Value Decomposition (rSVD), a fairly recent advancement from the field of randomized linear algebra, to speed up and parallelize the SVD as a core component of the SST. Combining these methods with a fast Hankel matrix product reduces the computational complexity of both methods from $\mathcal{O}(N^3)$ to $\mathcal{O}(N \log N)$. We demonstrate how these asymptotic complexities translate to runtime improvements for growing window sizes and provide theoretical arguments for our algorithmic choices. We substantiate these arguments with empirical measurements on an extensive dataset of real and synthetic time series. The main contributions of this paper can be summarized as follows:

- **SST and IKA-SST with log-linear time- and linear memory complexity.** We reduce the per-step computational complexity of the SST [2] and IKA-SST [3] from $\mathcal{O}(N^3)$ to $\mathcal{O}(q(k+p)N \log N)$, and reduce the space complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}((k+p)N)$. Over a length-L time series, this yields $\mathcal{O}(Lq(k+p)N \log N)$ (SST) and $\mathcal{O}(LkN \log N)$ (IKA-SST) with parameters $k, p, q \ll N$. This change in complexity results in faster computation times, and extends the applicability of both SST-variants to larger window sizes and high-frequency signals.
- **Suitability and Expected Error Bound.** By discussing the spectral properties of Hankel matrices and linking the scoring error with the decomposition error via a novel error bound, we provide theoretically grounded arguments for the suitability of the rSVD for general Hankel matrices, and SST-based CPD specifically.
- **Empirical Evaluation.** On extensive real and synthetic data, we confirm our theoretical claims, demonstrating low rSVD approximation errors and low end-to-end scoring errors. We demonstrate parallel scaling and strong runtime improvements that grow with N. For example, our improved IKA-SST is 814x faster at $N = 5000$. In a small case-study, our algorithmic modifications reduce the runtime from 22 hours to 22 minutes.
- **Artifacts.** We release reference implementations of the (improved) SST algorithms and our measurement code.

The remainder of this paper is structured as follows: We first introduce the related work. We explain our selection criteria for the related work (Sec. II), summarize their contributions and then distinguish our work from it. Following this, we discuss our methods (Sec. III), establishing all necessary concepts first (Sec. III-A & III-B). We explain both SST variants (Sec. III-C), and the rSVD (Sec. III-D) algorithm in detail and introduce a fast matrix product for Hankel matrices (Sec. III-E). We then show how the rSVD and the fast matrix product can be used to improve the SST and the IKA-SST respectively (Sec. III-F & III-G). In the last part of the methods section, we provide arguments for the suitability of the rSVD in terms of the approximation error (Sec. III-H), relying on spectral properties of the Hankel matrix and derive an error bound connecting the decomposition accuracy with the scoring error (Sec. III-I). In the evaluation section (Sec. IV), we introduce our dataset (Sec. IV-A) and provide implementation details (Sec. IV-B). Using the dataset, we empirically substantiate our assumptions on the spectral properties of Hankel matrices, measuring the singular value decay of Hankel matrices extracted from diverse time series (Sec. IV-C). We then empirically evaluate the approximation error of the rSVD (Sec. IV-D) and how it affects the final change score when built into the SST (Sec. IV-E). In this context, we also show how we derive several hyperparameters for the rSVD by employing other research on time series structure. In the same section, we empirically demonstrate that the fast Hankel matrix product does not introduce large numerical errors. As the last part of the evaluation section, we address our overall goal by evaluating how the reduced computational complexity translates to actual runtime improvements for all algorithms (Sec. IV-F). In the discussion (Sec. V), we reflect on the empirical results, discuss their implications, and the limitations of our improvement . Finally, we give an outlook on potential further improvements (Sec. V-B) and conclude the paper (Sec. VI).

To make our research reproducible and accessible, the source code for this paper and the datasets are available online[1].

[1]https://github.com/Lucew/approximate_hankel

While the repository contains the research code dedicated to recreating the contents of this paper, we also provide reference implementations of the CPD algorithms in the pip-installable Python package *changepoynt*[2]. By making the algorithms easily accessible, we aim to enable future applications to benefit from the proposed (IKA-)SST improvements.

II. RELATED WORK

This paper does not propose a new CPD algorithm. Instead, we improve an existing algorithm (SST [2]), and another variant of it (IKA-SST [3]). Therefore, we mainly discuss publications related our target algorithms, including methods based on Singular Spectrum Analysis (SSA) [6]. In addition to the related work on SST-based CPD, we discuss publications whose contributions involves the efficient decomposition of Hankel matrices, as we employ similar algorithms to improve SST-based CPD.

A. CHANGE POINT DETECTION

CPD methods aim to identify abrupt structural changes within time series. While there are different methods from statistics [7] and other domains [8]–[12], this paper focuses on techniques that employ SSA-based methods as discussed in beginning of this section. SSA-based methods extract time series characteristics by computing the SVD of matrices that contain subsequence sampled from a time series. The degree of change is estimated by comparing the extracted characteristics at two different times within the series. We refer to Section III-C for a detailed explanation on the SST. For a broader overview of time series CPD, see [1].

Moskvina et al. [6] build on the theory of SSA and compare the extracted characteristics with time series samples at a later point in time to detect changes. Idé et al. [2] published a variation of the algorithm termed SST, which they later improve for speed by adopting an implicit Krylov subspace approximation [3] (IKA-SST). These algorithms are widely accepted [1] and have been applied to different use cases [4], [5], [13]. The authors claim that IKA-SST is carefully designed to account for numerical errors caused by the subspace reduction but only show qualitative measurements to support that claim. Other, more recent improvements of the SSA-based CPD methods, mainly aim at improving the scoring function [5], [14], [15], or the application to multidimensional time series [16], rather than their computational efficiency.

Our paper builds on the related work on SSA-based change point detection, mainly addressing the runtime and computational complexity of the algorithms. While the IKA-SST is an already improved version of the SST, we improve it further. In our evaluation and discussion section (Sec. IV-E & V), we show how our improvements introduce only a small approximation error, arguing that our algorithmic modifications can be used as a drop-in replacement. Independent of our improvement, our data shows that the IKA-SST underestimates the score, so the SST remains a valid alternative.

[2]https://pypi.org/project/changepoynt/

B. EFFICIENT HANKEL MATRIX DECOMPOSITIONS

Hankel matrices and their decompositions are common in different areas of digital signal processing. While the goals in these areas may vary from those of CPD, we still want to consider their advancements. The following papers are related to our work as they share the SVD of a Hankel matrix and propose an efficient decomposition as part of their contribution. For a clearer distinction of our approach to the related work in this section, we want to reiterate that we apply two different algorithms from linear algebra to improve the SST and the IKA-SST. First, we employ the rSVD as a fast, truncated SVD. Second, we use a fast matrix product, exploiting the Hankel structure of our matrices.

Singular Spectrum Analysis. Korobeynikov [17] published an efficient decomposition of Hankel matrices with similar complexity as proposed in our paper. While their paper uses the structure of the Hankel matrix, the decomposition relies on a Lanczos bidiagonalization method, which is iterative in nature. The rSVD used in our paper supports parallelization, facilitating better utilization of modern, multicore CPUs. Still, we employ the Lanczos bidiagonalization as a baseline and show how the rSVD is comparable in accuracy, but is faster in practice. Furthermore, Korobeynikov [17] shows runtime measurements, but numerical errors and the method's applicability are neither discussed nor empirically measured. Rodrigues et al. [18] use randomized matrix decompositions to speed up SSA techniques for large time series but do not exploit the structure of the Hankel matrix, which allows a fast Hankel matrix product. Again, they mainly show time measurements and have limited to no discussion of errors.

Digital Signal Processing. Sahnoun et al. [19] use Lanczos bidiagonalization in combination with the structure of the (multilevel) Hankel matrix to arrive at a the same computational complexity as this paper. Still, the decomposition is iterative in nature, similar to [17]. Their paper also includes discussions and measurements of error, but only in the context of parameter estimation for dynamical systems, such as wireless communication channel estimation or antenna systems. Yang et al. [20] propose to use rSVD for signal denoising based on Hankel matrices. In contrast to our paper, they do not exploit the Hankel structure. Wang et al. [21] apply rSVD for large-scale system identification and also include a thorough error discussion focused on their specific application. Their application demonstrates that the Hankel matrices can quickly grow in size. In contrast to our paper, their approach does not use the Hankel structure or the parallelization inherent in the rSVD.

In general, all papers in this section differ from our research as they did not evaluate their methods in the context of SST-based time series CPD. We contribute theoretical arguments for the suitability of the rSVD in our CPD scenario and derive specialized error bounds. Furthermore, our contribution is not limited to the Hankel matrix rSVD for the SST, but we also improve the IKA-SST through the fast Hankel matrix product.

Algorithm 1 Singular Spectrum Transformation (SST) [2]. Final scoring function from [3].

```
Input: Time series $T \in \mathbb{R}^{1\times L}$, window size $N$, target rank $k$, lag $\delta$ between the Hankel matrices
Result: Series of scores $S \in \mathbb{R}^{1\times L}$
1: Initialize empty series of scores $S \in \mathbb{R}^{1\times L}$ with elements $S_i$
2: for $i = (2N-1+\delta)$ to $L$ do
3:     Create future Hankel matrix $H_f$ from the signal range $[i-(2N-1),\ i]$
4:     Compute the prominent singular vector $\mathbf{u}_f \in \mathbb{R}^{N\times 1}$ of future Hankel matrix $H_f$
5:     Create past Hankel matrix $H_p$ from the signal range $[i-(2N-1+\delta),\ i-\delta]$
6:     Compute the $k$ most prominent singular vectors $U_p \in \mathbb{R}^{N\times k}$ using past Hankel matrix $H_p$       } Replace with alg. 2
7:     Compute the score as one minus the norm of the matrix vector product $S_i = 1 - ||U_p^T\mathbf{u}_f||_2^2$   } for IKA-SST.
8: end for
9: return $S$
```

III. METHODS

Our goal is to increase the efficiency of both the SST and IKA-SST without introducing large approximation errors. In the following sections, we explain the necessary, basic concepts (Sec. III-A & III-B) to understand the SST and IKA-SST. We introduce both algorithms, and discuss their main computational bottlenecks (Sec. III-C). Subsequently, we explore the necessary methods to tackle these bottlenecks (Sec. III-D & III-E), and then explain how we use them to improve the SST and IKA-SST, respectively (Sec. III-F & III-G). Fig. 2 visualizes the algorithmic modifications to improve the SST efficiency. On a general level, different parts of the SST (alg. 1) can either be replaced by the IKA subroutine (alg. 2), or the rSVD (alg. 3). Both contain Hankel matrix multiplications, which can be replaced by a fast Hankel matrix product (alg. 4).

In addition to the algorithmic improvements, Section III-F discusses theoretical reasons for using the rSVD as part of the SST algorithm. On a high level, a truncated decomposition like the rSVD allows us to trade accuracy for speed. We examine this trade-off to show how the rSVD is well suited as a decomposition for Hankel matrices in terms of approximation errors (Sec. III-H), and derive an error bound that directly extends this arguments to the SST score (Sec. III-I).

The notation table in Appendix A provides an overview of the symbols used in this section.

A. HANKEL MATRICES

Hankel matrices H are special matrices with constant skew diagonals (e.g., fig. 1). While they may come up differently for other applications, they can be constructed from time series by filling the columns with subsequences using a sliding window over the original time series as visualized in fig. 1. For a window size N, the Hankel matrix contains $2N-1$ different samples overall. Note that the rows and columns represent sliding windows over the original time series, and the first row and last column together contain the complete time series used for constructing the matrix. A square Hankel matrix is symmetric, and, therefore, equal to its transpose ($H = H^T\ \forall\ H \in \mathbb{R}^{N\times N}$). Hankel matrices are closely related to Toeplitz matrices. Reversing the column order transforms a Hankel matrix into a Toeplitz matrix.

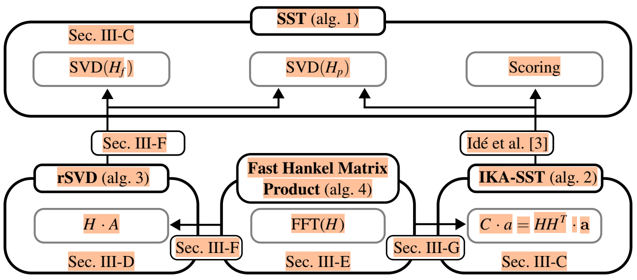


FIGURE 2: Visualization of the algorithmic interactions. Arrows indicate the replacement of an operation.

B. SINGULAR VALUE DECOMPOSITION

The SVD of a matrix SVD$(H) = U\Sigma V^T$ results in three matrices. U contains the left singular vectors in its columns, diagonal matrix Σ contains the singular values σ_i in decreasing order ($\forall i\ \sigma_i \geq \sigma_{i+1},\ i \in [1, min\{M,N\}]$), and V^T contains the right singular vectors as rows. In the case of a complete SVD, multiplying the matrices perfectly reconstructs the original matrix. The computational effort for the complete SVD depends on the matrix dimensions $\mathcal{O}(M*N*min\{M,N\})$, where M is the number of rows and N is the number of columns of the matrix [22]. For quadratic Hankel matrices with $M=N$, the complexity of the SVD is $\mathcal{O}(N^3)$.

Many applications only require the $k \ll N$ most prominent singular values and corresponding singular vectors U_k, Σ_k, and V_k. Therefore, a complete decomposition is often unnecessary. This gives rise to the development of partial decompositions that stop at k components and are referred to as truncated SVD algorithms.

We restrict the complexity terms and error bounds to $N=M$ for clarity, but all following algorithms also work for $N \neq M$.

C. SINGULAR SPECTRUM TRANSFORMATION

SSA-based CPD algorithms measure the degree of change in a time series by comparing the largest left singular vectors of two Hankel matrices. These matrices are constructed using time series samples from before and after some time point t. Intuitively, the largest singular vectors primarily capture the main characteristics of the time series used to construct the matrices, whereas the smaller ones mostly contain noise and minor variations. If these characteristics differ before and after t, a change has occurred.

In the scope of our paper, a time series is an ordered collection of tuples of length L, each consisting of a timestamp and a real signal value. We assume that the timestamps are equidistant from each other. A sliding window of length l returns l consecutive elements from the original time series.

On a high level, the SST algorithm [3] can be condensed into the steps shown in alg. 1. At every sample of a complete time series, two Hankel matrices, past and future, are constructed from the original time series T. The past Hankel matrix H_p contains time series samples before a point t, whereas the future Hankel matrix H_f contains samples after that point. Changes are detected by comparing the future Hankel matrix's most prominent left singular vector $\mathbf{u}_f$ to the k most prominent singular vectors U_p of the past Hankel matrix. Mathematically, the final score is computed as one minus the norm of the projection of $\mathbf{u}_f$ on the past singular vectors U_p ($||U_p^T \mathbf{u}_f||_2$, alg. 1, line 7).

Therefore, every SST-step requires two truncated decompositions of $N \times N$ Hankel matrices and a matrix vector product. With increasing window sizes, these decompositions quickly become the computational bottleneck of the algorithm.

The IKA-SST [3] is an improved version of the SST that aims to increase the speed of the original SST. The paper shows that the score as defined in alg. 1 can be computed in a subspace seeded by the future singular vector $\mathbf{u}_f$ without ever computing the past singular vectors U_p explicitly. The subspace is only slightly larger than the hyperparameter k, significantly smaller than N. See the steps in alg. 2 for details. This effectively removes the SVD of the large $N \times N$ Hankel matrix as a computational bottleneck. Although this dramatically reduces the runtime, the algorithm still requires the computation of the correlation matrix as a product of two matrices (alg. 2, line 1). A step that is embarrassingly parallelizable but still has cubic complexity $\mathcal{O}(N^3)$ concerning the matrix size N.

Note that we are using the updated scoring function $S_i = 1 - ||U_p^T \mathbf{u}_f||_2^2$ from the IKA-SST for both variants. The reasons are explained in the IKA-SST paper [3].

D. RANDOMIZED SINGULAR VALUE DECOMPOSITION

In their seminal paper, Halko et al. [22] present a modular framework for constructing randomized algorithms for truncated matrix decompositions, one of which is the randomized SVD (rSVD). The basic idea of rSVD is to create a smaller, representative subspace of the original matrix A by randomly sampling from its column space. The original matrix A is then projected onto this smaller subspace. Computing the SVD in this subspace is significantly faster due to the reduced dimensionality. Transforming the singular vectors from the subspace to the original space yields the final result. The suitability of the subspace, represented by the subspace projector Q, directly influences the projection error. Determining a suitable subspace is the focus of the rSVD algorithm.

The rSVD algorithm proposed by Halko et al. [22] can be seen in alg. 3. Lines 1-10 contain all operations to find a suitable Q. The first step is the random sampling from the column space using a random matrix $\Omega \in \mathbb{R}^{N\times(k+p)}$ to initialize the subspace (lines 1 & 2). An oversampling parameter p is used to enhance the precision of the decomposition by sampling $k + p$ columns. Projector Q should have orthonormal columns that represent most of the original space to make the projection as good as possible:

$$QQ^T \approx I \implies A \approx QQ^T A \tag{1}$$

Additionally, subspace iterations can be used to the enhance dominant singular values. The iterations decay the contribution of lower singular values by taking repeated powers of the input matrix. These power iterations are mixed into the creation of Q in the original paper (alg. 3, lines 4-9).

Due to the projection into a subspace with fewer dimensions, the complexity of the SVD in alg. 3 reduces to $\mathcal{O}((k+p)^2 N)$ and the bottleneck shifts to the q repeated matrix multiplications of the input matrix $A \in \mathbb{R}^{N\times N}$ with other matrices $\Omega, Q \in \mathbb{R}^{N\times(k+p)}$ for the sampling step and the subspace iterations. This keeps the overall complexity quadratic with respect to window size N ($\mathcal{O}(q(k+p)N^2)$). In contrast to other Krylov subspace methods, like Lanczos bidiagonalization, used in other publications that have iterative components, the matrix multiplications required to construct the subspace projector are embarrassingly parallelizable [22].

Algorithm 2 IKA-SST subroutine [3].

```
Input: Past Hankel matrix $H_p \in \mathbb{R}^{N\times N}$, target rank $k$,
    prominent future singular vector $\mathbf{u}_f \in \mathbb{R}^{N\times 1}$
Result: Approximated change point score $\hat{S}_i$
 1: Compute correlation matrix $C = H_p H_p^T \in \mathbb{R}^{N\times N}$
 2: Set Lanczos rank $\zeta = \begin{cases} 2k & k \in \text{even} \\ 2k-1 & k \in \text{odd} \end{cases}$
 3: Initialize $\alpha \in \mathbb{R}^{1\times(\zeta+1)} = 0$, $\beta \in \mathbb{R}^{1\times(\zeta+1)} = 1$
 4: Initialize $\mathbf{r} = \mathbf{u}_f$, $\mathbf{q} \in \mathbb{R}^{N\times 1} = 0$
 5: for $i = 1$ to $\zeta$ do                          ▷ Lanczos Subroutine
 6:     $\tilde{\mathbf{q}} = \mathbf{r}/\beta_{0,i}$
 7:     $\alpha_{0,i+1} = \tilde{\mathbf{q}}^T C \tilde{\mathbf{q}}$
 8:     $\mathbf{r} = C\tilde{\mathbf{q}} - \alpha_{0,i+1}\tilde{\mathbf{q}} - \beta_{0,i} * \mathbf{q}$
 9:     $\beta_{0,i+1} = ||\mathbf{r}||$
10:     $\mathbf{q} = \tilde{\mathbf{q}}$
11: end for
12: Tridiagonal matrix $\tilde{C} \in \mathbb{R}^{\zeta\times\zeta}$ with main diagonal set to
    $\alpha[1:]$ and off-diagonals set to $\beta[1:-1]$
13: Compute SVD$(\tilde{C}) = \tilde{U}\tilde{\Sigma}\tilde{V}^T$      ▷ Special Tridiag. SVD
14: $\hat{S}_i = 1 - \sum_{j=0}^{k-1} (\tilde{U}_{0,j})^2$  ▷ First k-elements of first row of $\tilde{U}$
15: return $\hat{S}_i$
```

E. FAST HANKEL MATRIX PRODUCT

While the rSVD can be applied to matrices of arbitrary structure, our matrices show Hankel structure due to the con-

Algorithm 3 Randomized Singular Value Decomposition (rSVD) with Subspace Iterations [22].

```
Input: Matrix $A \in \mathbb{R}^{N\times N}$, target rank $k$, oversampling
    parameter $p$, number of subspace iterations $q$
Result: Approximated, truncated SVD $(A) \approx \hat{U}_k\hat{\Sigma}_k\hat{V}_k^T$
 1: Generate a random matrix $\Omega \in \mathbb{R}^{N\times(k+p)}$.
 2: Form $Y_0 := A\Omega$                              ▷ Random Sampling.
 3: Compute QR factorization $Y_0 := Q_0R_0$ for initialization
 4: for $i = 1$ to $q$ do                           ▷ Subspace Iterations.
 5:     Form $\tilde{Y}_i := A^TQ_{i-1}$
 6:     Compute its QR factorization $\tilde{Y}_i = \tilde{Q}_i\tilde{R}_i$
 7:     Form $Y_i := A\tilde{Q}_i$
 8:     Compute its QR factorization $Y_i = Q_iR_i$
 9: end for
10: Save $Q = Q_i$
11: Form the matrix $B \in \mathbb{R}^{(k+p)\times N} := Q^TA$
12: Compute the SVD of the small matrix $B = \tilde{U}_k\hat{\Sigma}_k\hat{V}_k^T$
13: Project $\hat{U}_k := Q\tilde{U}_k$ back into original space
14: return $\hat{U}_k \in \mathbb{R}^{N\times k}$, $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$, and $\hat{V}_k \in \mathbb{R}^{N\times k}$
```

struction with sliding-window subsequences. Every column and row of the Hankel matrix is a sliding window of size N. Therefore, multiplying the matrix by a vector from the left or right is equivalent to the cross-correlation of the vector with the larger window of size $2N-1$. The naive matrix vector product of vector $\mathbf{a} \in \mathbb{R}^{N\times 1}$ with an arbitrary matrix $A \in \mathbb{R}^{N\times N}$ has a computational complexity of $\mathcal{O}(N^2)$.
Convolutions, particularly for larger kernels, can be efficiently computed in the frequency domain as noted in the convolution theorem [23]. The theorem states that the convolution in the time domain is a multiplication in the frequency domain. To convert the discrete cross-correlation into a discrete convolution, the vector has to be flipped (reverse element order). Due to the availability of the Fast Fourier Transform (FFT), the complexity of the multiplication of a Hankel matrix $H \in \mathbb{R}^{N\times N}$ with a vector $\mathbf{a} \in \mathbb{R}^{N\times 1}$ reduces to $\mathcal{O}(N \log N)$. Multiplying the Hankel matrix by any arbitrary matrix $A \in \mathbb{R}^{N\times(k+p)}$ can be reduced to multiple matrix vector products resulting in a complexity of $\mathcal{O}((k+p)N \log N)$. The multiplication of H by each of the $k+p$ columns of A is an independent operation and can therefore run in parallel.
Additionally, there is no need to construct the full Hankel matrix and allocate $\mathcal{O}(N^2)$ memory, which costs additional time and may cause memory problems for large window sizes. It is only ever necessary to compute the FFT for sliding windows of size $2N-1$, reducing the space complexity of the algorithm to a linear function of the window size $\mathcal{O}(N)$.

Alg. 4 lists the necessary steps to perform an efficient, aperiodic cross-correlation using the FFT. To multiply a Hankel matrix by an arbitrary matrix A, the algorithm basically performs several matrix vector products. The multiplication algorithm is not new and has been published several times before [17]. The algorithm iterates over the columns of

Algorithm 4 Fast Hankel Matrix Multiplication hankel_matmul.

```
Input: FFT representation of Hankel matrix
    $H \in \mathbb{C}^{1\times(2N-1)}$, arbitrary matrix $A \in \mathbb{R}^{N\times G}$ with
    columns $A_i$
Result: Multiplication result $R \in \mathbb{R}^{N\times G}$
 1: function hankel_matmul($H$, $A$)
 2:     Create empty matrix $R$ with columns $R_i$
 3:     for $i = 1$ to $G$ do         ▷ Iterations can run in parallel.
 4:         Flip the column $\hat{A}_i = A_i[:: -1]$ (and zero-pad)
 5:         FFT $\tilde{A}_i = \mathscr{F}(\hat{A}_i)$     ▷ padding: Sec. III-E & III-F
 6:         Multiplication $\tilde{R}_i = \tilde{A}_iH$              ▷ element-wise
 7:         Inverse FFT $\hat{R}_i = \mathscr{F}^{-1}(\tilde{R}_i)$
 8:         Extract column $R_i = \hat{R}_i[(N-1):(2N-1)]$
 9:     end for
10:     return $R$
11: end function
```

arbitrary matrix A, and performs the same, independent set of operations on each column of A. Beginning in line 4, we flip the column to account for the difference in cross-correlation and convolution, and zero-pad the column to the same size as the FFT-transformed Hankel matrix. We then perform the FFT on the column (line 5), and perform the multiplication in frequency space (line 6). After that, we apply the inverse FFT transform (line 7) and extract the elements from $N-1$ to $2N-1$ into the corresponding column of result matrix R to avoid circular convolution (line 8) [24].
The fast Hankel matrix product also applies to matrices that introduce a lag τ between the sliding time windows. In this case, the resulting matrix would not be a Hankel matrix by definition. However, we can account for these gaps. We do this by introducing τ zero-filled rows between all original rows of A, thereby increasing the number of rows from N to $N+(N-1)\tau$. The length of the time series in the Hankel matrix increases to $(2N-1)+(N-1)\tau$. In this case, the complexity for the matrix product results to $\mathcal{O}((k+p)\tau N \log(\tau N))$. The matrix vector product is, therefore, also applicable to matrices that do not strictly have Hankel structure but are similar in that the columns are subsequences of a time series. The speedup using the convolution theorem then depends on the value of τ.

F. IMPROVING SST THROUGH THE RSVD

We have now introduced all the necessary building blocks that we will use to improve the speed of the SST and IKA-SST. In short, we improve the SST (alg. 1) by replacing the SVD with the rSVD (alg. 3), and the Hankel matrix multiplications in the rSVD with the fast Hankel matrix product (alg. 4).
If the input matrix A to the rSVD (alg. 3) is a Hankel matrix, we can replace every product involving A with the fast Hankel matrix product (alg. 4). For preparation, we only have to transform the Hankel matrix into frequency space by applying the FFT to the entire time series of length $2N-1$ used to

construct the Hankel matrix (see Sec. III-A). The replacement of the naive matrix product directly addresses the bottleneck of the rSVD. The overall complexity of the rSVD then decreases from $\mathcal{O}(q(k+p)N^2)$ to $\mathcal{O}(q(k+p)N\log N)$. Alg. 5 (FFT rSVD) lists all steps of the specialized rSVD with detailed comments on the complexity of each operation. The hyperparameter k of the SST directly translates to the target rank k of the rSVD.

In addition to the main algorithmic adaptations discussed in the previous sections, we made several minor adaptations to the original algorithm. We substitute the QR-Factorization with the LU-Factorization as done in Li et al. [25] to further improve the runtime. Additionally, we zero-pad the original signal and use an FFT algorithm for real inputs. We zero-pad the vectors to reach an optimal length $\geq 2N-1$ for the divide and conquer FFT algorithm (provided by the FFT implementation).

The rSVD with the fast Hankel matrix product yields the following advantages over other decomposition methods:

- **Parallelization.** The term $(k+p)$, introduced to the computational complexity by the oversampling, is parallelizable over multiple cores in contrast to other, iterative methods. Improving the accuracy using the oversampling parameter p is not as costly in terms of processing time, as it can be offset by adding more workers.
- **Modifiable Error Bounds.** The upper error bounds are configurable by adapting the oversampling parameter p and the number of subspace iterations q.
- **Concise Implementation**. The implementation of the rSVD is straightforward, involving only a few steps and very common matrix operations. These are most likely readily available even on specialized hardware, such as GPUs and microprocessors.
- **Computational and Space Efficiency.** The computational complexity of the decomposition is only $\mathcal{O}(q(k+p)N\log N)$. Implemented in code, this yields significant speedup for the truncated decomposition of Hankel matrices compared with the complete SVD. Additionally, the space complexity has been reduced to $\mathcal{O}((k+p)N)$ from $\mathcal{O}(N^2)$, because there is no need to construct the full Hankel matrix $(k,p,q \ll N)$.

As we replace the SVD in the SST algorithm (FFT rSVD SST) with the specialized rSVD (FFT rSVD), all advantages of the rSVD transfer to the SST as well. In addition to the listed advantages, we argue that the rSVD will achieve low approximation errors for the Hankel matrices in our scenario. See Sections III-H and III-I for further details.

The standard SVD algorithm [26] overwrites input matrix structure to compute all components as accurately as possible. Therefore, we can not improve the complete SVD using the fast Hankel matrix product to replace the multiplications with the input matrix. Additionally, we only need the first few components, which is why we targeted truncated decompositions.

Note that our improvement of the rSVD is not limited to perfect Hankel matrices (lag of one), but only requires that the rows/columns are subsequences of one larger time series, sampled with some delay τ (see the end of Sec. III-E). Furthermore, this specialized rSVD version also applies to all matrices that are a product of Hankel matrices owing to the associativity of the multiplication. Observe that the input matrix A to the rSVD is only ever used in multiplications, so we never have to actually compute the product of these Hankel matrices. We will use a similar observation for improving the IKA-SST in the next section.

G. IMPROVING THE IKA-SST

For the IKA-SST, the implicit Krylov approximation shifted the bottleneck from the SVD to computing the correlation matrix C in line one of alg. 2. This multiplication is necessary to maintain a low approximation error, but has a complexity of $\mathcal{O}(N^3)$, scaling badly with the window size. To understand how to improve the IKA-SST, observe that the matrix C is only ever used directly in matrix vector multiplications (alg. 2, lines 7 & 8). By exploiting the associativity of the multiplication, we never have to compute C explicitly. We can skip line one and replace the matrix products (lines 7 & 8) with two fast Hankel matrix products $C\tilde{\mathbf{q}} = (HH^T)\tilde{\mathbf{q}} = H(H^T\tilde{\mathbf{q}})$. The product in brackets is a matrix vector product and the same is true for multiplying H by the resulting vector $H^T\tilde{\mathbf{q}}$ of the previous operation. Thus, we eliminate the computational bottleneck of the IKA-SST. The following sections will refer to this modified algorithm as FFT IKA-SST. Similar to the rSVD and SST, the space complexity reduces to $\mathcal{O}(N)$. This concludes our second objective of improving the IKA-SST. For readers interested mainly in the practical implications, we recommend jumping to Section IV-F ff, where we discuss the speedup.

H. DISCUSSION OF DECOMPOSITION ERRORS

Now that we have introduced our improvements to the SST and the IKA-SST, we want to provide further reasons for choosing the rSVD instead of other truncated decompositions, such as the implicitly restarted Lanczos bidiagonalization (IRLB) [27].

Summarized in Section III-F and discussed in detail by Halko et al. [22], the rSVD has several advantages over its competitors. As the main disadvantage they mention that, with the same computational budget, the rSVD is in many cases less accurate than, e.g., the IRLB. Nevertheless, we argue that the rSVD is comparably accurate to other sequential competitors in our special use case. To substantiate this claim, we discuss theoretical error bounds and how the spectral properties of time series Hankel matrices align conveniently with requirements derived from these bounds. Before doing so, we have to introduce general error bounds for both the truncated SVD in general and the rSVD in detail.

Minimal Truncation Error. Using all singular vectors and singular values computed by the SVD perfectly reconstructs the initial matrix. Truncation of the computed spectrum always introduces an error to the reconstruction. The absolute error depends directly on the power of the truncated (unused) singular values. The Eckart-Young-Mirsky theorem [28], [29]

Algorithm 5 Efficient truncated Hankel Decomposition based on rSVD.

```
Input: Time series $T \in \mathbb{R}^{1\times L}$, window size $N$, target rank $k$, oversampling parameter $p$, number of subspace iterations $q$
Result: Approximated, truncated Hankel SVD $(H) \approx \hat{U}_k \hat{\Sigma}_k \hat{V}_k^T$
1:  function hankel_rsvd($T, N, k, p, q$)
2:      // Create the Hankel representation (with padding to next best length $\geq 2N-1$)
3:      Extract the flattened Hankel matrix as a sliding window $\tilde{H} \in \mathbb{R}^{1\times(2N-1)}$ from $T$          ▷ $\mathcal{O}(N)$
4:      Compute the FFT $H := \mathscr{F}(\tilde{H})$ representation of the Hankel matrix          ▷ $\mathcal{O}(N\log N)$

5:      // Randomized sampling from column space
6:      Generate a random matrix $\Omega \in \mathbb{R}^{N\times(k+p)}$ with i.i.d. entries from a standard normal distribution.          ▷ $\mathcal{O}(N(k+p))$
7:      Form $Y_0 :=$ hankel_matmul($H, \Omega$)          ▷ $\mathcal{O}((k+p)N\log N)$

8:      // Begin of Subspace iterations with $Q \in \mathbb{R}^{N\times(k+p)}$
9:      Compute QR factorization $Y_0 = Q_0R_0$ for initialization          ▷ if $q > 1$: LU factorization [25].
10:     for $i = 1$ to $q$ do          ▷ Complete loop: $\mathcal{O}(q(k+p)N\log N)$
11:         Form $\tilde{Y}_i :=$ hankel_matmul($H, Q_{i-1}$)          ▷ alg. 4, $H^T = H$ (Sec. III-A)
12:         Compute its QR factorization $\tilde{Y}_i = \tilde{Q}_i\tilde{R}_i$          ▷ while $i < q$: LU factorization [25].
13:         Form $Y_i :=$ hankel_matmul($H, \tilde{Q}_i$)          ▷ alg. 4
14:         Compute its QR factorization $Y_i = Q_iR_i$          ▷ while $i < q$: LU factorization [25].
15:     end for

16:     // SVD in Subspace
17:     Form the matrix $B \in \mathbb{R}^{(k+p)\times N} :=$ (hankel_matmul($H, Q_q$))$^T$   ▷ $\mathcal{O}((k+p)N\log N)$, $Q_q^TH = (HQ_q)^T$, with $H^T = H$
18:     Compute the SVD of the small matrix $B = \tilde{U}_k\hat{\Sigma}_k\hat{V}_k^T$          ▷ $\mathcal{O}((k+p)^2N)$
19:     Project $\hat{U}_k := Q_q\tilde{U}_k$ back into original space          ▷ $\mathcal{O}(N(k+p)^2)$
20:     return $\hat{U}_k \in \mathbb{R}^{N\times k}$, $\hat{\Sigma}_k \in \mathbb{R}^{k\times k}$, and $\hat{V}_k \in \mathbb{R}^{N\times k}$
21: end function
```

states that a rank k reconstruction A_k using the first k singular vectors is better or equal to any other matrix with rank k in terms of the Frobenius ($||\bullet||_F$) and spectral l_2 norm ($||\bullet||_2$) (for a decreasingly sorted singular spectrum). The error of the truncated SVD reconstruction of rank k can be computed as follows [28], [29]:

$$||A - A_k|| = \begin{cases} \sigma_{k+1} & \text{for the } ||\bullet||_2 \text{ norm} \\ \left(\sum_{i=k+1}^{r} \sigma_i^2\right)^{1/2} & \text{for the } ||\bullet||_F \text{ norm} \end{cases} \quad (2)$$

$$||A - X|| \geq ||A - A_k|| \text{ for all } X \text{ with } \mathrm{rank}(X) = k \quad (3)$$

In other words, the error caused by the truncation is directly related to the singular value spectrum of the matrix A and the decay therein. The faster the spectrum decays, the better is the truncated reconstruction. For our analysis of errors in this section, we will concentrate on the $l2$ norm similar to Halko et al. [22].

rSVD Approximation Error. The rSVD relies on random sampling, so the error bounds can only be formulated as an expected error $\mathbb{E}^*$ over the realizations of the sampling step. The error is introduced mainly by the subspace projection in line 11 of alg. 3. Halko et al. [22, Corollary 10.10] arrive at the following error bounds for this step (with Euler's number e, target rank k, oversampling p, subspace iterations q, and matrix size N):

$$\mathbb{E}^*_{\text{Proj}} := \mathbb{E}||A - QQ^TA||_2 \quad (4)$$

$$\mathbb{E}^*_{\text{Proj}} \leq \underbrace{\left[1 + \sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p}\sqrt{N-k}\right]^{\frac{1}{2q+1}}}_{=:f^k_{proj}} \sigma_{k+1} \quad (5)$$

$$\mathbb{E}^*_{\text{rSVD}} \leq \sigma_{k+1} + \mathbb{E}^*_{\text{Proj}} \quad (6)$$

$$\text{with q} \sim \log N: \ \mathbb{E}^*_{\text{Proj}} \sim \sigma_{k+1} \implies \mathbb{E}^*_{\text{rSVD}} \sim 2\sigma_{k+1} \quad (7)$$

Note that we replaced the tail term $(\sum_{j>k}\sigma_j^{2(2q+1)})^{1/2}$ with its largest term $\sqrt{N-k}\cdot\sigma^{2q+1}$ as done in the original paper [22, Corollary 10.10 f]. This is pessimistic but makes clear how the truncation error of the rSVD is connected to the SVD. When comparing (6) with (2), we see that the rSVD introduces an additive error to the minimal possible value of σ_{k+1}. This additive term quantifies the fidelity of the subspace projection and decreases with an increasing oversampling parameter p and even exponentially with an increasing number of subspace iterations q. When q approaches the vicinity of $\log N$, the additional error is only the largest dropped singular value [22]. In addition, we should note that the size of the matrix N increases the error. The upper bound is described as overly pessimistic in the original publication but gives an analytical expression that shows the essential factors for the method's applicability. The equation shows that the subspace projection works best for matrices with a rapidly decaying singular spectrum and, therefore, a small σ_{k+1}. Consequently, replacing the computationally expensive SVD with the truncated rSVD works best for matrices with this property.

General Hankel Matrices. Intuitively, Hankel matrices from non-chaotic time series should have a fast-decaying singular spectrum, because the columns are constructed using a sliding time window over the time series. Since the windows largely overlap, it is likely that the columns are highly correlated, and the resulting Hankel matrix is of low rank. In other words, only a few vectors are necessary to reconstruct the matrix. This intuition is formalized in [30]. The paper notes that linear recurrence relations in the time series are a necessary and sufficient condition for rank-deficient Hankel matrices. Based on these findings, we argue that the rSVD is suitable for the truncated decomposition of Hankel matrices.

SST-based Hankel Matrices. When using the rSVD as part of the SST, the low-rank assumption becomes a necessary precondition for even applying the SST algorithm. The SST, an SSA-based algorithm [2, Sec. 5], is mostly limited to sufficiently structured signals, resulting in low-rank Hankel matrices [6, Sec. 6.2]. In cases where time series have stochastic or chaotic patterns, even the naive SST algorithm is not expected to deliver good detection results. We argue that, in these cases, where the naive SST is unsuitable, additional decomposition errors are not the primary concern. Consequently, we can expect low-rank Hankel matrices when replacing the SVD with the rSVD in the SST algorithm.

Positive Semi-Definite Hankel Matrices. As noted in the related work Section II, the decomposition of Hankel matrices is a common part of many algorithms for time series and system analysis. Therefore, we want to add some notes on positive semi-definite Hankel matrices. We cannot guarantee this property in our scenario, but our insights may be of interest to others. See Appendix D for further details.

I. CONNECTING DECOMPOSITION AND SCORING ERROR

From the previous section, it is clear that the rSVD is a suitable candidate for decomposing (SST-based) Hankel matrices. Nevertheless, it remains unclear how exactly the decomposition error influences the final SST score (Sec. III-C). The score depends mainly on the extracted singular vectors (alg. 1, line 7), which are not part of the error bounds in the previous section. Intuitively, an accurate decomposition yields accurate singular vectors, which in turn yield an accurate score, however, the exact mechanism remains unclear. Although our evaluation will show that this intuition is correct, this section analyzes this connection in detail. The goal of this section is to derive an analytical error bound that directly connects the rSVD decomposition error to the SST scoring error.

By interpreting the projection residual $||(A - QQ^TA)||$ (see (4)) as an effective perturbation $||E||_2$ to an arbitrary SVD-input matrix A, we can use matrix perturbation theory [31] to derive a bound for the scoring error of the SST, analytically connecting the decomposition error of the rSVD with the scoring error. At a high level, matrix perturbation theory, specifically Wedin's theorem [32], quantifies how perturbations of A propagate to its singular values and associated singular vectors. We define the scoring error E_S as follows:

$$E_S := S_i - \hat{S}_i = (1 - ||U_p^T \mathbf{u}_f||_2^2) - (1 - ||\hat{U}_p^T \hat{\mathbf{u}}_f||_2^2) \\ = ||\hat{U}_p^T \hat{\mathbf{u}}_f||_2^2 - ||U_p^T \mathbf{u}_f||_2^2 \tag{8}$$

After some modifications to the equation and relying on Wedin's theorem [32], we arrive at an upper bound on the scoring error that determined by the perturbation, i.e. the quality of the subspace projection. When replacing the SVD with the rSVD in the SST, there are two sources of error. The errors are introduced by the decomposition of H_p (E_p) and H_f (E_f), respectively. For the derivation, refer to Appendix B:

$$|E_S| \le 4\frac{||E_p||_2}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{||E_f||_2}{\sigma_{f,1} - \sigma_{f,2}} \tag{9}$$

This bound can be used for any alternative decomposition. In our case, the perturbation $||E||_2$ is the projection error $\mathbb{E}^*_{\text{Proj}} = ||A - QQ^TA||_2$ of the rSVD written in (4). As this error is an expected value, our scoring error also transforms into an upper bound of the expected scoring error. Finally, we can tie the scoring error to the rSVD decomposition error:

$$\mathbb{E}\left[|E_S|\right] \le 4\frac{\mathbb{E}^*_{\text{Proj},p}(k,p,q,N)}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{\mathbb{E}^*_{\text{Proj},f}(p,q,N)}{\sigma_{f,1} - \sigma_{f,2}} \tag{10}$$

Thus, a smaller decomposition error directly yields a smaller expected scoring error. By inserting (4) into (10), we see that increasing the parameter p and q also reduces the upper bound on the expected scoring error in the same way they reduce the decomposition error. Using the approximation $q \sim \log N \rightarrow \mathbb{E}^*_{\text{Proj}} \sim \sigma_{k+1}$ [22], the bound reduces to:

$$\mathbb{E}\left[|E_S|\right] \lesssim 4\frac{\sigma_{p,k+1}}{\sigma_{p,k} - \sigma_{p,k+1}} + 4\frac{\sigma_{f,2}}{\sigma_{f,1} - \sigma_{f,2}} \tag{11}$$

Equation (9) is independent of the decomposition method, but we use the rSVD with a Gaussian random projector, allowing us to employ specialized bounds for randomized decompositions that bound the error with high probability (derivation in Appendix C):

$$|E_S| \le 2\frac{\sigma_{f,2}}{\sigma_{f,1}}\frac{\left(\frac{\sigma_{f,2}}{\sigma_{f,1}}\right)^{2q}}{1 - \frac{\sigma_{f,2}}{\sigma_{f,1}}}\left(\frac{1}{\sqrt{p-1}} + \frac{e\sqrt{(1+p)(N-1)}}{p}\right) \\ +2\frac{\sigma_{p,k+1}}{\sigma_{p,1}}\frac{\left(\frac{\sigma_{p,k+1}}{\sigma_{p,k}}\right)^{2q}}{1 - \frac{\sigma_{p,k+1}}{\sigma_{p,k}}}\left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p}\right) \tag{12}$$

Both, (10) and (12), demonstrate how a fast-decaying singular spectrum, enlarging $\sigma_k - \sigma_{k+1}$ and decreasing σ_{k+1}, directly translates to a low expected scoring error. Equation (10) shows that the scoring error is a sum of the decomposition errors for the past- and future Hankel matrices scaled by the singular value gap $\sigma_j - \sigma_{j+1}$.

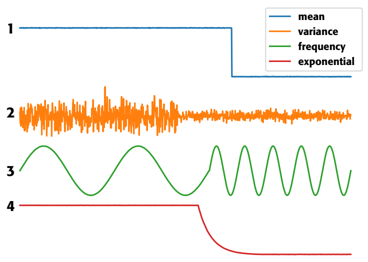


FIGURE 3: Examples for each type of simulated signals with randomized change points.

Note that the bounds are conservative and pessimistic. In practice, when the leading singular values are very close to each other (i.e., the Hankel matrix does not have a clear dominant direction, sometimes the case stochastic or weakly structured signals), the spectral gap is small and the theoretical bound on the score error becomes weak, because the bound depends inversely on the spectral gaps $(\sigma_j - \sigma_{j+1})$.

With our bound, we mathematically prove that a fast-decaying singular spectrum of the Hankel matrix leads to a low scoring error, when we replace the SVD with the rSVD in the SST algorithm. Fortunately, we expect the Hankel matrices to exhibit a fast-decaying singular spectrum. This spectral property is likely for general Hankel matrices, and even a precondition for the Hankel matrices occurring in the SST (Sec. III-H). This concludes our theoretically grounded argument on the suitability of the rSVD for SST-based CPD: The rSVD drastically improves the computational efficiency of the SST at the cost of only a small scoring error.

IV. EVALUATION

In the methods Section III, we discussed the specialized rSVD and the fast Hankel matrix product, and then used these two building blocks to improve the SST and the IKA-SST. During the discussion of the error, we showed that a fast-decaying singular spectrum results in a small approximation error (see (4)) and also directly in a small scoring error (see (10)). This theoretical evidence supports the applicability and suitability of the rSVD for Hankel matrices. Now we want to substantiate the claims with empirical measurements using a collection of simulated and real world time series. We empirically measure the singular value spectrum, decomposition quality, and scoring error. Last but not least, we measure algorithmic runtime to evaluate whether we reached our goal of speeding up SST-based CPD. Owing to the variety of measurements and amount of data, the total computation time for all experiments was approximately ten days.

Our goal is to accelerate SST-style scoring to enable scalable CPD, not to tune detectors for specific datasets. Threshold dependent metrics (e.g., F1) vary with application-specific post-processing and beyond scope of our evaluation. See the last part of the discussion Section V for further comments.

A. DATASETS

We rely on two different datasets, with real and simulated time series for our evaluation.

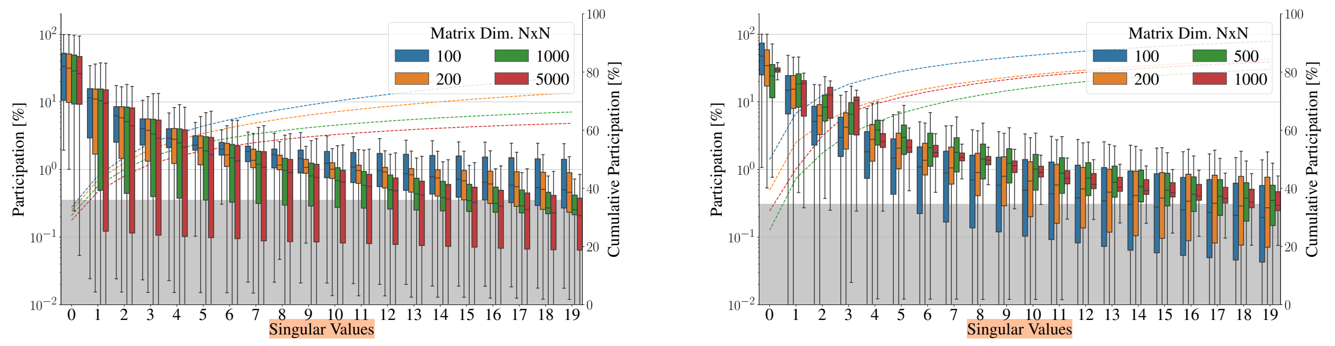
Synthetic Signals. We simulate four different types of changes. Examples of each type are shown in figure 3. Each type has two random parameters, the first one being the position of the change and the second one being the magnitude of the change. For the mean change, the magnitude is the difference between the two plateaus, and both are chosen to be a random integer percentage between zero and one. The second type is a change in the variance of two normal distributions, where the variances are chosen in a similar manner to the mean change. The third one is a change in the frequency of a sine wave with two frequencies chosen similar to the mean and variance. The fourth is an exponential decline, in which the slope of the decline is chosen randomly. Our defined types of changes are in line with the related work on CPD [3], [7]. We create this dataset to ensure that our measurements contain time series with change points.

Real Signals. For our real-world time series, we rely on the diverse UCR Time Series Classification Archive [33]. At the time of this publication, the dataset contains 191,158 signals of varying lengths from 256 different datasets. We excluded 218 signals from the dataset due to missing values.

B. IMPLEMENTATION DETAILS

The methods are implemented in Python 3.11 using linear algebra operations provided by Numpy (v1.26) [34]. Additionally, we use Numba (v0.59) [35] as a JIT compiler and for parallelization. Generally, all mathematical operations used are common. Therefore, an implementation on specialized hardware like GPUs and microprocessors is straightforward. While error measurements were run on both the Intel i9-10980XE and AMD EPYC 7713 processor, the timing measurements were performed only on the AMD processor. Both systems were running Ubuntu 24.04 with 64 GiB and 512 GiB main memory (DDR4@3200 MHz RAM) for the Intel and AMD processors, respectively. The implementation is optimized on a feasible level, but specialized lower-level languages may further reduce the runtime of our algorithm. However, the purpose of the measurements in this paper is to show the relative advantages of the methods, more than optimizing our code for the fastest absolute runtime.

Typically, the SST operates as a sliding window over the signals, processing largely overlapping time series windows. This implies highly correlated results for the scoring error measurements. To counteract this, we compute our results on non-overlapping parts of the signals. For the real-world dataset, we cut the signals in non-overlapping sections to construct the Hankel matrices. Consequently, the number of evaluated matrices decreases as the window size grows. For example, a signal with 400 samples is cut into two separate matrices for a window size of $N = 100$ (note that the entire Hankel matrix spans a larger window of size $2N - 1$). For the synthetic signals, we simulate each signal with the exact length required to construct one Hankel matrix.



(a) Synthetic Signals. (b) Real Signals.

FIGURE 4: The singular spectrum of synthetic and real signals for different window sizes N. The values are normalized by the sum of all singular values from each matrix. The secondary axis on the right shows the cumulative participation. The gray region in the background marks the threshold for unknown noise proposed by Gavish et al. [36] around which we cut off the plot. A fast spectral decay benefits rSVD approximation error.

TABLE 1: Matrix sizes used for singular value spectrum analysis.

Synthetic Signals	100	200	1000	5000
Real Signals	100	200	500	1000

C. SINGULAR VALUE SPECTRUM

The first research question we are trying to answer is whether the rSVD is an appropriate method for the decomposition of Hankel matrices constructed from time series. The error bounds in Section III-H are mainly determined by the first excluded singular value in the truncated decomposition, as shown in (6). Therefore, we compute the complete singular value spectrum of all non-overlapping Hankel matrices extracted from our datasets for different matrix sizes (see tab. 1). For the synthetic dataset, we simulated 500 different Hankel matrices for each window size and four different signal types, resulting in 2000 singular values per boxplot, shown in fig. 4. For the real signals, we chose 1000 as the largest matrix size, because most signals in the UCR archive were shorter than 2000 samples. From the real dataset, we gathered 303057 matrices for the smallest window and 2535 for the largest window. The signals have different value ranges. Because the value range directly influences the absolute magnitudes of the singular values, we normalize each singular value by the sum of all singular values of the complete decomposition.

The resulting singular value spectra are visualized in fig. 4. Each color corresponds to a different matrix size. Each boxplot shows the distribution of the nth singular value. In addition to the boxplots, dashed lines show the cumulative percentage of singular values in the sum of all singular values. The results show an evident exponential decay of the singular value spectrum. This empirically confirms the suitability of the rSVD for Hankel matrices. Consequently, we also expect a low decomposition and scoring error.

D. DECOMPOSITION ERROR

To quantify the decomposition error E_D (see (13)), we compare the reconstructed matrix with the original matrix. As an optimal baseline reconstruction, we use the singular vectors computed by the complete SVD. We compare this baseline with the reconstruction error of the singular vectors from both the rSVD and the rSVD using the fast matrix product (FFT rSVD). E_D is the difference between the reconstruction errors by the SVD and rSVD, and normalized by the SVD reconstruction error to make the results comparable for varying signal value ranges:

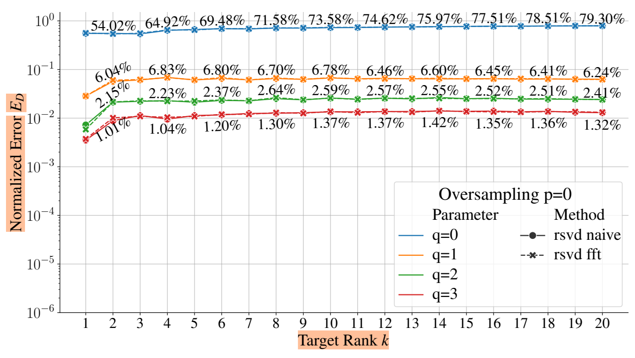
$$E_D(k,p,q) := \frac{\left|\left|A - A_k^{\text{rSVD}(p,\,q)}\right|\right|_F - \left|\left|A - A_k^{\text{SVD}}\right|\right|_F}{\left|\left|A - A_k^{\text{SVD}}\right|\right|_F} \tag{13}$$

The matrix sizes are the same as those in the previous section (see tab. 1). Apart from varying the target rank k, we computed the results for different oversampling parameters p and various numbers of subspace iterations q.

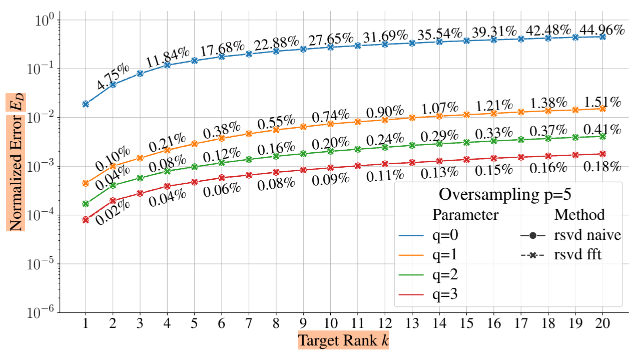
The results in fig. 5 behave as expected when compared with the theoretical bounds discussed in the previous section on the rSVD error (see (4) ff.). In general, the error introduced by the rSVD is fairly small, underlining the suitability of the rSVD for Hankel matrices (Sec. III-H). Oversampling achieves good improvements. This is interesting, because the additional operations for the oversampling can be offset to some degree by parallelization. The computationally expensive subspace iterations yield the highest improvement in terms of decomposition error. Comparing the dashed and solid lines in fig. 5, we see that the fast Hankel matrix product introduces no significant error.

E. SCORING ERROR

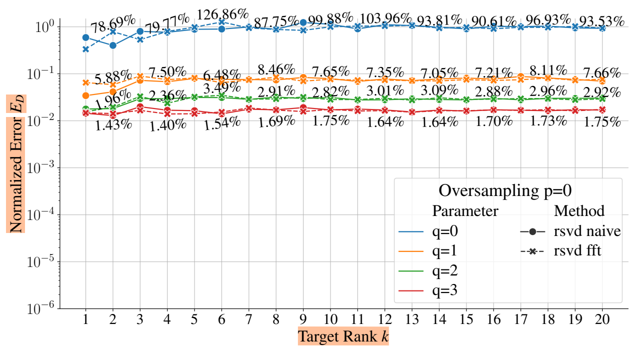
After evaluating the decomposition error, we now turn to the scoring error for the SST. The scoring is different from the reconstruction of the original matrix, as it compares the left singular vectors of two different matrices. To compute the error, we compare the different algorithms with the naive



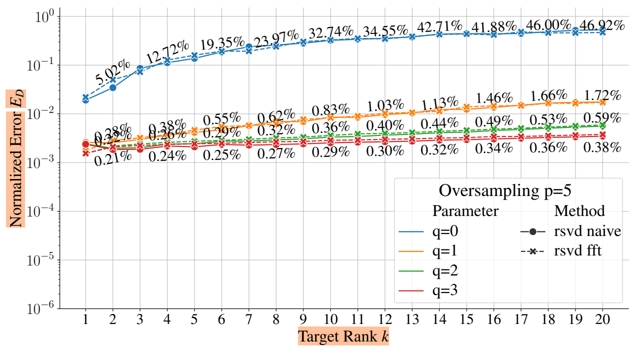
(a) Synthetic signals with oversampling $p = 0$.



(b) Synthetic signals with oversampling $p = 5$.



(c) Real signals with oversampling $p = 0$.



(d) Real signals with oversampling $p = 5$.

FIGURE 5: Decomposition error E_D (see (13)) for different target ranks k. Lower is better. The error is plotted on a logarithmic axis. Compare left and right plots for different oversampling parameters p. Compare upper and lower plots for different datasets.

implementation that uses the complete SVD for both decompositions (alg. 1, lines 4 & 6) as a baseline. Additionally, we use the implicitly restarted Lanczos bidiagonalization (IRLB) by Baglama and Reichel [27] for comparison. The IRLB is a state-of-the-art truncated decomposition with a computational complexity similar to the rSVD. Other than the rSVD it is sequential, and harder to stabilize numerically [22]. We implemented it as an alternative for computing the truncated decomposition. The IRLB implementation has been modified from an open-source repository [37]. The scoring error E_S for each method is computed as the difference between the exact score S_i and the approximated score $\hat{S}_i$ (see (8)).
Without the fast Hankel matrix product, we compare three ways to speed up the baseline algorithm:

- The IKA-SST proposed by Idé et al. [3] (IKA).
- The SST with the IRLB for the decompositions (IRLB).
- The SST with the rSVD for the decomposition (rSVD).

Each of the three variants can use the fast Hankel matrix product (FFT). Therefore, we finally evaluate six different SST variants, and the naive SVD baseline. Tab. 2 provides an overview of the computational complexities of all variants including the baseline.

Selection of Hyperparameters. Several hyperparameters must be specified for each of the variants. For all variants, we set the target rank to $k = 5$. The value is a hyperparameter from the SST and IKA-SST that does not originate from our algorithmic adaptions. SSA-based CPD papers set k to values below ten [2], [3], [6]. As a guideline they mention that k should be small enough to capture all important components, but large enough to ignore minor variations. They also mention that it should be based on the singular value spectrum, setting k so that the corresponding singular values are relatively large. As we use the rSVD, k additionally influences the computational effort and the approximation error (tab. 2 & (12)). The empirical singular value spectrum (Sec. IV-C) shows that for $k = 5$ the participation drops below 5% for all signals and window sizes reliably (within one standard deviation), with a cumulative participation well over 50%. Therefore, we argue that a value of $k = 5$ is suitable for capturing the main structure and small enough to only consider main components, but it remains a hyperparameter of the SST. This value is within the range of values used in the original publications [2], [3], [6]. Furthermore, all methods contain parameter k with the same meaning, allowing relative comparisons of the methods with a fixed k. We evaluated the effect of k on the decomposition in the previous section, and our bound shows that the effect translates to the score in a similar way (Sec. III-I).
For the IRLB, we have to specify the estimation tolerance and a maximum number of iterations. We keep these parameters to the default values of 10^{-4} for the estimation tolerance and a maximum of 50 iterations. One could further decrease

the scoring error of the IRLB with these parameters, but this would increase the runtime and the results show that the IRLB is already slower than the IKA-SST and rSVD SST with these parameters, while not being significantly more accurate.
The rSVD also requires two additional parameters: the oversampling value p and the amount of subspace iterations q. These parameters directly affect the expected error (see (6)). Based on the recommendation by Halko et al. [22], we set the number of subspace iterations to $q \approx \log N \approx 3$, as the maximum window sizes for our measurement are around 10^3 (see (10) & (12)).
Next is the oversampling parameter p. For the evaluation in this paper, we used three different orientation methods. We implemented two rank identification models for parametric time series models reviewed by Golyandina [38], namely the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). In addition, we computed the threshold for signal denoising from Hankel matrices proposed in the seminal paper of Gavish et al. [36]. To find an appropriate $k + p$, we computed the three thresholds over all matrices used in the previous section (Sec. IV-C). The mean values for all three rank estimation methods are listed in tab. 3. Based on this empirical observation, we set the oversampling parameter p so that $k + p = 15$. Our reasoning is that all further singular values after 15 will likely contain only noise, which is irrelevant for shape-based CPD.
Results. Similar to Sections IV-C and IV-D, we compute the scoring error on matrices of different sizes extracted from both datasets. In contrast to the previous evaluation, two different matrices have to be extracted before and after a certain point t (see alg. 1). For the synthetic dataset, we created 72000 matrix pairs spread over 30 different window sizes on a log scale ranging from 100 to 5000. From the real signals, we extracted 5.498.270 matrix pairs spread over 83 different window sizes on a log scale ranging from 100 to 1169 for each decomposition method. The mean absolute errors are consolidate in tab. 4. Their distribution is shown in fig. 6.
Tab. 4 clearly shows that using the fast Hankel matrix product (FFT) introduces no significant scoring error. Overall, the accuracy of our rSVD SST is comparable to the IRLB SST and slightly better than the IKA-SST for the given parametrization. Tab. 4 also shows that the scoring error is notably different between real and synthetic signals. We attribute this difference to the distribution of change scores between the two datasets. While the synthetic data aims to introduce deliberate changes via the simulation, the real signals show no changes more frequently.
In the histogram (fig. 6), the most notable difference between the methods is the high frequency of positive change score errors when using the IKA-SST (on the right tail). A positive error implies that the baseline detects a high change score, while the approximated method reports a lower score. In other words, "there are a few peaks which are not reproduced by IKA" [3, Sec. 5.2]. Our results empirically reproduce the qualitative statement in the original IKA-SST paper on a larger dataset as a by-product of our evaluations. Comparing the FFT IKA-SST with the IKA-SST in tab. 4 and fig. 6, the results also show that this difference originates from the original algorithms, and not from our modifications.
The difference in IKA-SST and SST scoring is not directly within the scope of our evaluation, because the methods are proposed elsewhere [2], [3]. Focusing on our specific contributions, we mainly compare scores of the IKA-SST with the FFT IKA-SST, and the SST with the FFT rSVD SST, not the SST with the IKA-SST. With our measurements we aim to demonstrate our algorithmic speed improvements of SST and IKA-SST, while quantifying the accuracy-speed trade-off, aiming for a low scoring error. Nevertheless, our quantitative measurements allow us to clearly visualize the difference in SST and IKA-SST, which might be interesting to practitioners, as it is only qualitatively discussed in the original paper [3]. Fig. 7 clearly visualizes the scoring difference by plotting the approximated scores as distributions over the ground truth scores. It shows how the IKA-SST decouples from the true score for higher values. The other methods mostly follow the optimal line. As previously shown, this behaviour of the IKA-SST is not introduced by our modifications.

TABLE 2: Computational complexity of SST variants with time series length L, window size N, target rank k, number of Lanczos iterations g, oversampling parameter p, and number of subspace iterations q. Typically $g \sim k$ for convergence of the IRLB decomposition.

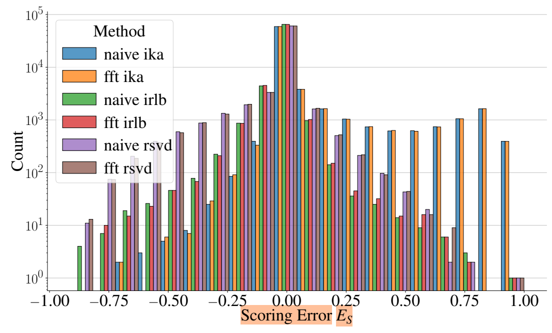
Method	Time Complexity	Corresp. Sec.
Naive SST [2]	$\mathcal{O}\left(LN^3\right)$	II-A
IKA-SST [3]	$\mathcal{O}\left(LN^3\right)$	II-A
SST + rSVD	$\mathcal{O}\left(Lq(k+p)N^2\right)$	III-F
SST + IRLB	$\mathcal{O}\left(LgN^2\right)$	IV-E
SST + rSVD + FFT	$\mathcal{O}\left(Lq(k+p)N\log N\right)$	III-F
SST + IRLB + FFT	$\mathcal{O}\left(LgN\log N\right)$	IV-E
IKA-SST + FFT	$\mathcal{O}\left(LkN\log N\right)$	III-G

TABLE 3: Mean values of BIC, AIC, and Noise Threshold measured using all datasets.

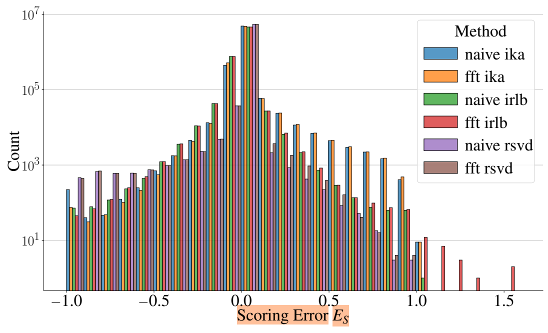
Dataset	BIC [38]	AIC [38]	Noise Threshold [36]
Synthetic Signals	11.86	14.03	16.53
Real Signals	10.56	15.16	17.00

TABLE 4: Mean absolute scoring error of SST variants relative to SVD baseline.

Scoring Error	Synthetic Signals	Real Signals
Naive IKA [3]	71.74×10^{-3}	9.695×10^{-3}
FFT IKA (this paper)	71.63×10^{-3}	9.672×10^{-3}
Naive IRLB [27]	16.83×10^{-3}	8.071×10^{-3}
FFT IRLB [39]	16.89×10^{-3}	8.124×10^{-3}
Naive rSVD [22]	36.20×10^{-3}	1.239×10^{-3}
FFT rSVD (this paper)	35.95×10^{-3}	1.392×10^{-3}



(a) Synthetic Signals.



(b) Real Signals.

FIGURE 6: Scoring error E_S distribution. Computed as the approximated score subtracted from the exact score $S_i - \hat{S}_i$ (see (8)). The error is positive when the approximation underestimates the score ($\hat{S}_i < S_i$) and negative for an overestimation ($\hat{S}_i > S_i$.)

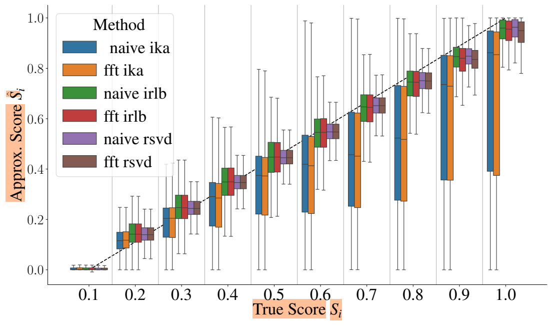


FIGURE 7: Approximated scores over the exact scores for the real signal dataset. The dashed line shows perfect equivalence.

F. RUNTIME SCALING AND PARALLEL EFFICIENCY

We discussed the theoretical complexity improvements for both the SST and IKA-SST in previous sections (Sec. III-F & III-G). Nevertheless, the algorithmic complexities only quantify asymptotic behaviour. We are interested in reducing the SST runtime. Consequently, we have to measure how the asymptotic improvement transfers to the practical application of the algorithms. We evaluate the runtime for our improved versions in comparison to the normal versions of the SST and IKA-SST. Recall that we improved the SST by replacing the complete SVD with our specialized rSVD (FFT rSVD SST, Sec. III-F) and the IKA-SST by never explicitly computing the correlation matrix and employing the fast Hankel matrix product (Sec. III-G).

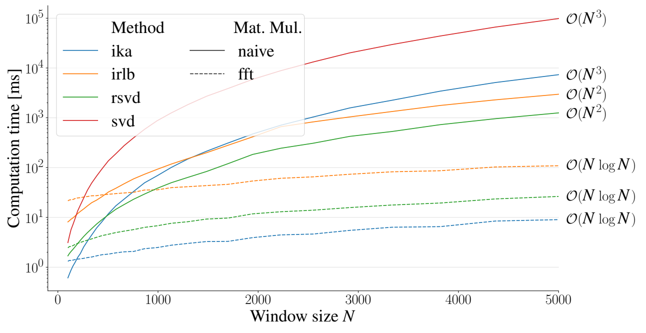
The absolute execution time will vary with the processor, programming language, and other system parameters (Sec. IV-B). Consequently, our measurements are mainly meaningful in relation to each other. For absolute timings, the implementation may have further optimization potential, which is beyond the scope of this paper. The runtime does not depend on time series structure. The execution times were measured on the AMD processor for the same matrix sizes as in the previous Section IV-E. We use the simulation from Sec. IV-A to create the matrices. To quantify the advantages of parallelization, we also vary the amount of threads.

Runtime Scaling. The main results of the runtime measurement are shown in fig. 8. The runtime is measured for one step of the loop in alg. 1. For both plots, the maximum number of threads was fixed to ten.

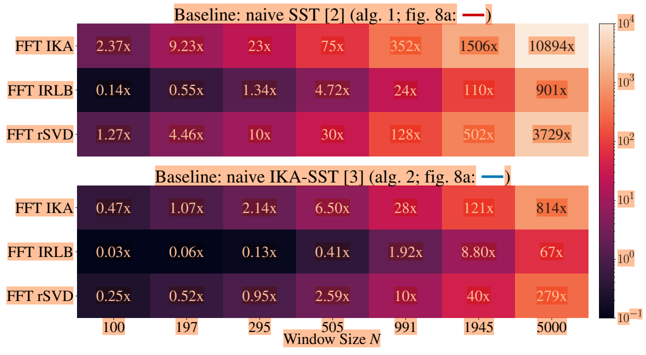
Fig. 8a shows the absolute runtime for different window sizes N and allows us to compare all variants within one plot. The fastest method, our FFT IKA-SST, overtakes all other methods approximately $N = 200$. The FFT rSVD SST is the second fastest method. At a window size of $N \approx 300$, it overtakes even the original IKA-SST. Until these points, the naive computations are faster, mostly because the FFT is a costly operation. The constant FFT cost materializes in the offset at the beginning of each dashed line. Comparing the IRLB with the rSVD, it can be seen that that the rSVD is faster. The plot also shows the algorithmic advantage of employing the fast Hankel matrix product. For example, the specialized FFT rSVD (dashed green) is 300x faster than the naive rSVD (solid green) at $N = 5000$. For large window sizes, the slope of the lines allow to clearly distinguish the algorithmic complexities (logarithmic axis). Due to special, parallel matrix multiplications for smaller matrices, the slopes of lines representing the naive algorithms take some time to settle, while the slopes are nearly constant for the improved SST versions.

Fig. 8b consolidates our runtime improvements by comparing our proposed versions of the SST and IKA-SST with the original algorithms. To compute the runtime factors, we divide the the runtime of the original algorithm runtime by that of the improved variant. Improving the IKA-SST with the fast Hankel matrix product (FFT IKA-SST), yields the largest runtime improvement. The reduction of the original $\mathcal{O}(N^3)$ complexity to $\mathcal{O}(N \log N)$, manifests in a speedup of 814x for a window size of 5000. Compared to the naive SST, we see a speedup of more than four order of magnitude (10894x). Compared to the naive SST, both the FFT IKA-SST and the FFT rSVD SST are faster for all window sizes.

Parallelization. Tab. 5 lists the timings for the IRLB, the rSVD and the SVD for a window size of 5000 and different thread numbers. With respect to the parallelization, using ten threads for the rSVD results in 40 % of the execution time



(a) Execution times of the different methods for one SST step. Lower times are better.



(b) Improved- versus naive SST (fig. 8a: ··· vs. —). Numbers $>$ 1x indicate that the method on the left is faster than the baseline.

FIGURE 8: Comparing the scalability of the SST with different algorithmic components (10 threads).

TABLE 5: Execution times of iterative IRLB SST, rSVD SST and naive SVD SST for different thread limits ($N = 5000$).

Threads	**FFT rSVD** (ours)	**FFT IRLB**	**SVD** (seconds!)
1	66.04 ms	112.79 ms	99.56 s
2	42.81 ms	105.81 ms	99.43 s
4	31.50 ms	108.50 ms	98.36 s
6	27.93 ms	107.74 ms	98.05 s
8	25.54 ms	108.73 ms	98.18 s
10	26.34 ms	108.98 ms	98.22 s

TABLE 6: Pairwise execution time comparisons among $\mathcal{O}(N \log N)$ SSTs for different thread limits. A value > 1 indicates that the method in the denominator is faster.

Threads	$\frac{\text{FFT rSVD}}{\text{FFT IKA}}$	$\frac{\text{FFT IRLB}}{\text{FFT IKA}}$	$\frac{\text{FFT IRLB}}{\text{FFT rSVD}}$
1	5.40	15.00	2.99
2	4.02	15.00	3.94
4	3.11	14.95	4.95
6	2.79	15.07	5.54
8	2.63	14.87	5.78
10	2.52	15.05	6.19

compared to single-threaded execution. This parallelizability is a significant advantage of the rSVD compared to other truncated methods, such as the IRLB, especially on modern hardware. The parallelization yields diminishing returns and ceases to decrease after eight threads. Parallelization is only possible over the $k + p$ columns of the subspace matrix. A suitable rule of thumb could be one thread per two columns (we set $k + p = 15$ for our measurements). However, this may be vary with the implementation details (Sec. IV-B).

Comparison of Efficient Variants. Finally, we compare the runtime efficiency of the three $\mathcal{O}(N \log N)$ methods: the FFT rSVD SST, the FFT IKA-SST, and the FFT IRLB SST. For this purpose, we create comparison factors by dividing the execution time of the slower method by the execution time of the faster method and average this value for all window sizes, creating pairwise runtime factors. Tab. 6 presents these runtime factors. Standard deviations were $\leq$ 0.25 for all window sizes and thread limits.

We clearly see that our FFT IKA-SST is the fastest method, but computing the more accurate rSVD SST comes close, when multithreading is allowed. For ten threads, the IKA-SST is only approximately $2.5\times$ faster. Looking at the last column, it is again notable how the rSVD benefits from parallelization when compared with the IRLB.

In summary, the experiments clearly show that the theoretical improvements transfer to the actual implementation. The methods are significantly faster for growing window sizes, while our algorithmic choices do not cause major approximation errors.

G. APPLICATION EXAMPLE

It is beyond the scope of this paper, to evaluate the SST for CPD. Nevertheless, we want to give a quick demonstration of how our improvement impacts the application of the SST. For our use case, we turn to the MIT-BIH Arrhythmia Database [40], which is a common dataset for medical signal processing. It contains the Electrocardiogram (ECG) recordings of several patients. As common in biomedical signals, the shape of the signal is important for the medical assessment, necessitating relatively high sampling rates. The signals are sampled with 360 Hz and are 30 minutes long, containing approximately 650 000 samples per recording.

For brevity, we focus on the IKA-SST and only one patient (patient 234, lead V1), where the signal contains two annotations of changing signal quality ($\sim$) sourced from [41]. To detect these, several heartbeats must be incorporated within the SST detection window. Based on initial tests, we selected 10 heartbeats. In the case of a normal resting heartbeat at 60 bpm, this requires a window size of $N = 360 * 10/2 = 1800$, as the sliding window of the SST is of size $2N - 1$. We run our experiments on a standard Google Colab instance (Intel Xeon Platinum 8259CL CPU @ 2.50GHz) and compute the change score at every fifth sample of the signal.

While the original IKA-SST requires more than 22 hours of computations, our improved version only requires around 22 minutes, running faster than the recording itself. This substantially reduces analysis runtime for applications building on the scoring results and allows online scoring even on commodity hardware.

V. DISCUSSION AND OUTLOOK

A. DISCUSSION

We set out with the initial goal of improving the SST and IKA-SST in terms of speed and scaling with the window size. We rely on the rSVD and the fast Hankel matrix product for these improvements and argued that the rSVD as a truncated decomposition is especially suitable (Sec. III-F & III-H). In our evaluation, we empirically demonstrated the following:

1) Section IV-C: Time series Hankel matrices have a fast decaying singular spectrum.
2) Section IV-D: The rSVD achieves a low approximation error for time series Hankel matrices.
3) Section IV-D: How we chose the parameters of the rSVD and how they affect the approximation error.
4) Section IV-E: In comparison with other competitors, the scoring error of the FFT rSVD SST is fairly small.
5) Section IV-E: The IKA-SST underestimates the score, but this is not due to our algorithmic modifications.
6) Sections IV-D and IV-E: Using the fast matrix product introduces no significant errors.
7) Section IV-F: The asymptotic complexities translate to runtime improvements at fairly small window sizes. The acceleration factors only increase with the window size.
8) Section IV-F: Parallelizing the FFT rSVD SST improves runtime compared to the sequential IRLB.
9) Section IV-F: The FFT IKA-SST is the fastest algorithm (for window sizes ≥ 200, given our implementation).
10) Sections IV-D and IV-F: The FFT rSVD SST out-scales the original IKA-SST, while being more accurate.

Overall, the evaluations support our theoretical arguments and hypotheses. We conclude that the rSVD is a suitable candidate for improving the naive SST. It is faster than other sequential decompositions, such as the IRLB, while maintaining a comparable accuracy and supporting parallel operations. The FFT IKA-SST is the fastest of all methods, but the scoring suffers from the same flaw as the original IKA-SST: it sometimes underestimates the change score. However, this flaw is independent of our improvements. If the SST-based CPD was suitable before, the FFT SST and the FFT IKA-SST are drop-in replacements. For growing window sizes, our modified versions are significantly faster, and only incur a minor approximation error, as expected from the theory and shown in the evaluation.

We recommend the FFT IKA-SST when a low execution time is the highest priority and the FFT rSVD SST, for the best speed-accuracy trade-off. Although the improved IKA-SST is the fastest available method, there are several reasons for using the rSVD SST. It is slightly slower, but the scaling is equal to the IKA-SST, and the scoring is significantly more accurate. Furthermore, the IKA-SST is restricted to computing the change score as the similarity of the largest future singular vector with the singular vectors of the past Hankel matrix. Other methods for computing the change score, such as subspace angles between future and past Hankel matrix, require more than the largest future singular vector, which IKA does not allow. The rSVD SST does not have this restriction and allows other scoring functions, e.g., [14], [15].

The fast matrix product is key in reaching our final asymptotic complexity and introduces no significant numerical errors. However, the involved FFT is expensive and the main reason for the constant offset in the runtime measurements. This offset determines the window size beyond which the proposed algorithms out-scale their original counterparts. The offset is clearly visible in fig. 8a. The dashed lines are initially higher than their solid counterparts. With our limited optimizations, we already reached a reasonably small window size, where the improvements overtake. While we chose Python for reproducibility and its ecosystem, implementing the algorithms in a speed-optimized language closer to the hardware could further reduce this offset. Section V-B lists some ideas to further improve the efficiency independent of the implementation language. Owing to the FFT representation, we never have to construct the complete Hankel matrix, which also saves time. Acquiring memory and filling the Hankel matrix scales with $\mathcal{O}(N^2)$. The linear space complexity was not our initial concern, but is a welcome by-product.

Halko et al. [22] mention that the rSVD with the same computational budget is less accurate than, e.g., IRLB, in most applications. Our empirical results suggest that this difference is almost negligible for time series Hankel matrices. We argued that this is mainly because of their spectral properties as discussed in Section III-H and measured in Section IV-D. In combination with the numerical stability, configurable error bounds, and parallelization, we argue that the advantages of the rSVD outweigh minor differences in accuracy. Nevertheless, the IRLB remains a valid alternative. The rSVD used in this paper is not the only method to compute the truncated SVD using randomized subspaces. There are other, more recent methods using block Krylov approximations and different subspace extractions [42], [43]. We decided on the rSVD as it is well understood in terms of decomposition errors, and the implementation is straightforward.

Although we illustrate our methods using square Hankel matrices $H \in \mathbb{R}^{N\times N}$ for clarity, the same approaches directly extend to rectangular Hankel matrices $H \in \mathbb{R}^{M\times N}$ that arise in many SST-based CPD variants. In this case, the FFT-based Hankel products and randomized decompositions are unchanged; replace N by $\min\{M, N\}$ in the complexity terms and error bounds to account for rectangular matrices.

There are also algorithms that perform SSA-based CPD on multivariate signals. In some cases this results in multilevel Hankel matrices, e.g. [16]. The fast Hankel matrix product can be applied to multilevel Hankel matrices, by performing several independent matrix multiplications for each contained Hankel matrix [19]. Therefore, alg. 5 can also be a adapted to multilevel Hankel matrices. The discussion of the decomposition error for multilevel Hankel matrices are beyond the scope of this paper, but our results indicate that the Hankel matrix product does not introduce large, numerical errors into the rSVD, which should remain the case for multilevel Hankel matrices.

Our improvements address the computational bottleneck of SST and IKA-SST. However, they do not alter the inherent domain of applicability of these methods. SST-based CPD is particularly effective for structured signals that exhibit repeating patterns, seasonality, or quasi-stationary regimes, where Hankel matrices display a rapidly decaying singular spectrum [6]. In contrast, stochastic signals or signals with chaotic dynamics often lead to slowly decaying spectra, reducing the effectiveness of low-rank approximations and degrading the quality of SST-based change scores. Partly this can be offset by increasing p and q, but both induce additional computational effort. In such cases, alternative methods may be preferable. Examples include relative density estimation [7], Bayesian Online CPD [44], Gaussian process CPD [45], and parametric methods [46]. Our contributions should be understood as enabling scalable SST-based CPD within its natural regime, rather than serving as a universal CPD solution. In practice, the choice between SST and alternative detectors should be guided by the signal structure and detection philosophy.

Nevertheless, the SST and IKA-SST are widely applied algorithms in different scenarios. Our contribution ensures that both are faster for growing window sizes and can now be applied in high-frequency or long-horizon scenarios where they were previously computationally prohibitive.

B. OUTLOOK

Our analysis of the FFT rSVD concentrates mainly on its applicability for time series CPD. As mentioned in Section II, there are further potential applications for the FFT rSVD in system identification or dynamical systems analysis. We also provide some thoughts on a cheap online estimation of errors in the case of positive semi-definite Hankel matrices that are outside of the scope of our paper (Sec. III-H). Detailed parametrization of the method, i.e., q and p for the rSVD, as well as numerical error introduced by the FFT, need to be evaluated for each application separately. As noted in Section III-F, the FFT rSVD extends to any matrix that is a product of Hankel matrices when the generating Hankel matrices are known. Ye et al. [47] prove that any matrix can be constructed as a product of Hankel matrices. In the case of a small number of Hankel factors, the fast Hankel matrix product may speed up the computation. Unfortunately, there is no known method to efficiently construct these Hankel factors [48].

There is literature on how to improve the fast Hankel matrix product [49]. Potentially, their ideas could decrease the overhead of the FFT as the main step in the fast matrix product, further decreasing the threshold after which the improved algorithms out-scale the original versions. For the complete scoring of a time series, the SST repeatedly computes separate decompositions on largely overlapping signal parts. This may leave room for further improvements. For example, sliding-window FFTs could potentially further decrease the runtime, e.g., using the algorithm in [50]. This idea of a sliding-window FFT directly interconnects with the application of the SST to streamed time series. Naively, the SST can be used on streaming time series by collecting the values in a rolling buffer of size $2N-1$, repeating lines 3-7 of alg. 1 for newly arriving samples. With this setup, it is crucial that the algorithms run faster than newly arriving samples. Our application example (Sec. IV-G) shows that our improvements enable buffered processing even for larger window sizes. With a sliding-window FFT, the runtime may decrease further, because the FFT would not process the complete window repeatedly. We consider these aspects for future work.

VI. CONCLUSION

Due to the involvement of SVD, the per-step runtime of the SST [2] cubically grows $\left(\mathcal{O}(N^3)\right)$ with the window size N. While the IKA-SST [3] eliminates the SVD bottleneck, the cubic complexity remains. Consequently, SST-based applications are limited to small window sizes or forced to implement extensive preprocessing steps, adding further hyperparameters to the pipeline. For long-duration patterns or high-frequency signals, applications often require larger window sizes to span the entire region of interest.

We tackle this scaling problem by utilizing the rSVD [22] and a fast Hankel matrix product to reduce the computational complexity of both methods to $\mathcal{O}(N\log N)$, and the space complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Except for minor approximation errors, our algorithmic modifications do not alter the detection outcome, but they significantly reduce the algorithmic runtime as the window sizes increases.

To substantiate our algorithmic choices, we provide theoretical arguments for the suitability of the rSVD for SST-based CPD. By discussing the fast spectral decay of Hankel matrices and linking the decomposition error with the scoring error via a novel error bound, we demonstrate why the more efficient rSVD incurs only a small approximation error, when replacing the SVD in the SST.

Empirical measurements on a large body of real and synthetic time series confirm our claims: the expected properties hold, score error remains small, and the asymptotic gains translate into wall-time improvements at moderate N. Our modified variants out-scale the naive SST, and even the IKA-SST. We measure runtime improvements of up to two and three orders of magnitude for IKA-SST (814x) and SST (3729x), respectively (fig. 8). Applying the accelerated IKA-SST to detect quality changes within an ECG signal demonstrates that our proposed modifications reduce detection time from several hours to mere minutes (22 h $\rightarrow$ 22 min, Sec. IV-G). We release the code for our experiments and provide reference implementations to facilitate further applications.

Among the accelerated variants, the FFT IKA-SST is the fastest method, while the FFT rSVD SST offers the best trade-off in terms of speed and accuracy (Sec. V).

In summary, our algorithmic modifications substantially improve the scalability of the SST and IKA-SST. They enable significantly larger window sizes within the same computational budget, reducing the runtime of current applications, and extending the applicability of the SST-based CPD to high-frequency or long-horizon scenarios.

REFERENCES

[1] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, pp. 339–367, 5 2017.

[2] T. Idé and K. Inoue, "Knowledge discovery from heterogeneous dynamic systems using change-point correlations," in *Proceedings of the SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 5 2005, pp. 571–575.

[3] T. Idé and K. Tsuda, "Change-point detection using krylov subspace learning," in *Proceedings of the SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 5 2007, pp. 515–520.

[4] L. Weber and R. Lenz, "Machine learning in sensor identification for industrial systems," *it - Information Technology*, vol. 65, pp. 177–188, 2023.

[5] S. Wu *et al.*, "Online adaptive anomaly detection in networked electrical machines by adaptive enveloped singular spectrum transformation," *IEEE Internet of Things Journal*, 2024.

[6] V. Moskvina and A. Zhigljavsky, "An algorithm based on singular spectrum analysis for change-point detection," *Communications in Statistics - Simulation and Computation*, vol. 32, pp. 319–352, 5 2003.

[7] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 5 2013.

[8] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh, "Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels," in *IEEE International Conference on Data Mining (ICDM)*, vol. 2017-November. IEEE, 5 2017, pp. 117–126.

[9] T. D. Ryck, M. D. Vos, and A. Bertrand, "Change point detection in time series data using autoencoders with a time-invariant representation," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3513–3524, 5 2021.

[10] N. Keriven, D. Garreau, and I. Poli, "Newma: A new method for scalable model-free online change-point detection," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3515–3528, 2020.

[11] A. Ermshaus, P. Schäfer, and U. Leser, "Clasp: parameter-free time series segmentation," *Data Mining and Knowledge Discovery*, vol. 37, pp. 1262–1300, 5 2023.

[12] G. Romano, I. A. Eckley, and P. Fearnhead, "A log-linear nonparametric online changepoint detection algorithm based on functional pruning," *IEEE Transactions on Signal Processing*, vol. 72, pp. 594–606, 2024.

[13] J. Yu, Y. He, Q. Yan, and X. Kang, "Specview: Malware spectrum visualization framework with singular spectrum transformation," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5093–5107, 2021.

[14] Y. Mohammad and T. Nishida, "Robust singular spectrum transform," in *International Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems*. Springer, 2009, pp. 123–132.

[15] Y. Mohammad and T. Nishida, "On comparing ssa-based change point discovery algorithms," in *2011 IEEE/SICE International Symposium on System Integration*. IEEE, 2011, pp. 938–945.

[16] A. Alanqary, A. Alomar, and D. Shah, "Change point detection via multivariate singular spectrum analysis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 218–23 230, 2021.

[17] A. Korobeynikov, "Computation- and space-efficient implementation of ssa," *Statistics and Its Interface*, vol. 3, pp. 357–368, 2010.

[18] P. C. Rodrigues, P. G. S. E. Tuy, and R. Mahmoudvand, "Randomized singular spectrum analysis for long time series," *Journal of Statistical Computation and Simulation*, vol. 88, pp. 1921–1935, 5 2018.

[19] S. Sahnoun, K. Usevich, and P. Comon, "Multidimensional esprit for damped and undamped signals: Algorithm, computations, and perturbation analysis," *IEEE Transactions on Signal Processing*, vol. 65, pp. 5897–5910, 5 2017.

[20] Y. Yang and J. Rao, "Robust and efficient harmonics denoising in large dataset based on random svd and soft thresholding," *IEEE Access*, vol. 7, pp. 77 607–77 617, 2019.

[21] H. Wang and J. Anderson, "Large-scale system identification using a randomized svd," in *American Control Conference (ACC)*. IEEE, 5 2022, pp. 2178–2185.

[22] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, pp. 217–288, 5 2011.

[23] T. K. Boehme and R. Bracewell, "The fourier transform and its applications." *The American Mathematical Monthly*, vol. 73, p. 685, 5 1966.

[24] T. G. Stockham Jr, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, 1966, pp. 229–233.

[25] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert, "Algorithm 971: An implementation of a randomized algorithm for principal component analysis," *ACM Transactions on Mathematical Software*, vol. 43, pp. 1–14, 5 2017.

[26] J. Dongarra *et al.*, "The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale," *SIAM review*, vol. 60, no. 4, pp. 808–865, 2018.

[27] J. Baglama and L. Reichel, "Augmented implicitly restarted lanczos bidiagonalization methods," *SIAM Journal on Scientific Computing*, vol. 27, pp. 19–42, 5 2005.

[28] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 5 1936.

[29] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The Quarterly Journal of Mathematics*, vol. 11, pp. 50–59, 1960.

[30] J. Gillard and K. Usevich, "Hankel low-rank approximation and completion in time series analysis and forecasting: a brief review," *Statistics and Its Interface*, vol. 16, pp. 287–303, 2023.

[31] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*. London: Academic Press, 1990.

[32] P.-Å. Wedin, "Perturbation bounds in connection with singular value decomposition," *BIT Numerical Mathematics*, vol. 12, no. 1, pp. 99–111, 1972.

[33] H. A. Dau *et al.*, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, pp. 1293–1305, 5 2019.

[34] C. R. Harris *et al.*, "Array programming with numpy," *Nature*, vol. 585, pp. 357–362, 5 2020.

[35] S. K. Lam, A. Pitrou, and S. Seibert, "Numba," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. ACM, 5 2015, pp. 1–6.

[36] M. Gavish and D. L. Donoho, "The optimal hard threshold for singular values is $4/\sqrt{3}$," *IEEE Transactions on Information Theory*, vol. 60, pp. 5040–5053, 5 2014.

[37] B. W. Lewis and M. Kane, "Irlbpy," https://github.com/bwlewis/irlbpy, 2018, accessed: 03.06.2024.

[38] N. Golyandina, "Particularities and commonalities of singular spectrum analysis as a method of time series analysis and signal processing," *WIREs Computational Statistics*, vol. 12, no. 4, p. e1487, 2020.

[39] K. Browne, S. Qiao, and Y. Wei, "A lanczos bidiagonalization algorithm for hankel matrices," *Linear Algebra and its Applications*, vol. 430, pp. 1531–1543, 5 2009.

[40] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[41] A. Roman, H.-M. D. of Health Sciences, Technology, R. G. Mark, and G. B. Moody, "Mit-bih arrhythmia database," https://www.kaggle.com/datasets/protobioengineering/mit-bih-arrhythmia-database-modern-2023, 2025, accessed: 08.08.2025. [Online]. Available: https://www.kaggle.com/dsv/12526755

[42] C. Musco and C. Musco, "Randomized block krylov methods for stronger and faster approximate singular value decomposition," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 1, 5 2015, pp. 1396–1404.

[43] M. F. Kaloorazi and R. C. de Lamare, "Subspace-orbit randomized decomposition for low-rank matrix approximations," *IEEE Transactions on Signal Processing*, vol. 66, pp. 4409–4424, 5 2018.

[44] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[45] Y. Saatçi, R. D. Turner, and C. E. Rasmussen, "Gaussian process change point models," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 927–934.

[46] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[47] K. Ye and L.-H. Lim, "Every matrix is a product of toeplitz matrices," *Foundations of Computational Mathematics*, vol. 16, pp. 577–598, 5 2016.

[48] K. Ye, "New classes of matrix decompositions," *Linear Algebra and Its Applications*, vol. 514, pp. 47–81, 5 2016.

[49] K. Ye and L.-H. Lim, "Algorithms for structured matrix-vector product of optimal bilinear complexity," in *IEEE Information Theory Workshop (ITW)*. IEEE, 9 2016, pp. 310–314.

[50] Z. Rafii, "Sliding discrete fourier transform with kernel windowing," *IEEE Signal Processing Magazine*, vol. 35, pp. 88–92, 5 2018.

[51] S. O'Rourke, V. Vu, and K. Wang, "Random perturbation of low rank matrices: Improving classical bounds," *Linear Algebra and its Applications*, vol. 540, pp. 26–59, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0024379517306420
[52] A. K. Saibaba, "Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms," *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 1, pp. 23–48, 2019.
[53] B. Beckermann and A. Townsend, "On the singular values of matrices with displacement structure," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, pp. 1227–1248, 5 2017.

REPRODUCIBILITY

The code and data used to create this paper are publicly available (https://github.com/Lucew/approximate_hankel). The reference implementations are available through the *changepoynt* package hosted on PyPi (https://pypi.org/project/changepoynt/). You can install them using the command *pip install changepoynt*.

ACKNOWLEDGMENT

The authors would like to thank Prof. Eamonn Keogh and everyone who contributed to the UCR time series archive, used in this paper. We also thank the editor and the reviewers for their valuable comments and suggestions. SIEMENS Energy AG supported this work in the context of Project DARTS. We thank Martin Buehl and Markus Zenker for their valuable insights into industrial data analytics.

LUCAS WEBER studied Electrical Engineering and Information Technology at the Technical University of Dresden. He is currently a research assistant at the chair of Evolutionary Data Management in the Computer Science Department of the Friedrich-Alexander Universität Erlangen-Nürnberg. His research interests include pattern recognition and signal processing for robust machine learning systems as well as knowledge discovery and mining in time series.

RICHARD LENZ is Professor for Evolutionary Data Management in the Department of Computer Science at the Friedrich-Alexander Universität Erlangen-Nürnberg. He leads research on evolutionary information systems, data quality and integration, workflow management, and research data management. He received his Ph.D. from the Friedrich-Alexander Universität Erlangen-Nürnberg with a thesis on "Adaptive Data Replication in Distributed Systems" and his habilitation at the University of Marburg for his work on "Evolutionary Information Systems in Healthcare".

APPENDIX. A

We use dedicated notation to separate special variables throughout the paper. For better readability, we list the most important symbols that are not standard notation in the following table. We introduce all symbols in the paper itself. It is self-contained without this overview.

Special Notations	Explanation
SST & IKA-SST	
Matrix H	Hankel matrix introduced in Section III-A.
Matrix C	Correlation matrix $C = HH^T$.
Matrix T	Time Series.
Matrix S	Score Series (the same size as T).
Vectors $\alpha, \beta, \mathbf{q}, \mathbf{r}$	Intermediate vectors of the IKA-SST (alg. 2).
Scalar N	Window size of the SST. Equal to the window size for the SVD matrices (see Sec. III-C).
Scalar k	Truncation rank, a main hyperparameter of the SST. It also occurs in the IKA-SST (alg. 2) and the rSVD (algs. 3 & 5), as they are used in the SST.
Scalar L	Time series length.
Scalar l	Sliding-window size.
Scalar S_i	The score of a single SST step (alg. 1, line 7).
Scalar t	A point in time.
Scalar δ	Time gap between the two Hankel matrices (alg. 1).
Scalar τ	Time gaps between sliding windows.
Scalar ζ	Lanczos rank in the IKA-SST (alg. 2).
SVD & rSVD	
Matrix U	Matrix containing the left singular vectors.
Matrix Σ	Matrix containing the singular values in decreasing order on the main diagonal.
Matrix V	Matrix containing the right singular vectors.
Matrix Q	The subspace projector of the rSVD (Sec. III-D).
Matrix Ω	The sampling matrix of the rSVD (Sec. III-D).
Vector $\mathbf{u}$	A left singular vector.
Scalar σ	A singular value. Singular values are always sorted in decreasing order ($\sigma_i \geq \sigma_j \, \forall i \leq j$).
Scalar k	Truncation rank. Equivalent to k from the SST section, as the rSVD is used as part of the SST.
Scalar p	Oversampling parameter of the rSVD (Sec. III-D).
Scalar q	Number of rSVD subspace iterations (Sec. III-D).
Errors	
Symbol E	General character to denote an error term.
Scalar E_S	Scoring Error (see (8)).
Scalar E_D	Decomposition Error (see (13)).
Scalar E_f	Decomposition Error of the future Hankel matrix H_f.
Scalar E_p	Decomposition Error of the past Hankel matrix H_p.
Scalar $\mathbb{E}^*_{\text{Proj}}$	Decomposition Error of the rSVD (see (4), notation taken from Halko et al. [22, Corollary 10.10 f]).
Sub- & Superscripts	
Subscript $\square_p$	Denotes variables that are linked to the past Hankel matrix H_p (alg. 1).
Subscript $\square_f$	Denotes variables that are linked to the future Hankel matrix H_f (alg. 1).
Subscript $\square_k$	Denotes truncated results of the SVD or rSVD.
Superscript $\hat{\square}$	Denotes approximations.
Superscript $\tilde{\square}$	Denotes intermediate results.
Miscellaneous	
Matrix R	Result of the fast Hankel matrix product (alg. 4).
Matrices A, X	Arbitrary matrices with no special structure.
Matrices B, Y	Intermediate results in the rSVD (alg. 3).
Vector $\mathbf{a}$	Arbitrary vector with no special structure.
Scalar G	Arbitrary matrix size in alg. 4
Symbol $\mathscr{F}(\square)$	Fast Fourier Transform (FFT).
Symbol $[a : b]$	Extract values a to b from a vector or matrix of size $1 \times \square$.

APPENDIX. B

We now provide the proof of (9) in Section III-I. The equation ties the decomposition error to the final scoring error of the SST, providing a clean connection between the employed decomposition and the final scoring error.
Notations and general statements for the following equations:

- $H_p,\ H_f \in \mathbb{R}^{N\times N}$ are the past/future Hankel matrices
- $U_p \in \mathbb{R}^{N\times k}$ contains the top-k orthonormal left singular vectors of H_p
- $\mathbf{u}_f \in \mathbb{R}^{N\times 1}$ is the top left singular vector of H_f
- $\sigma_{p,i}/\sigma_{f,i}$ are the singular values of H_p/H_f sorted in descending order ($\sigma_{\cdot,i} > \sigma_{\cdot,i+1} \forall i$)
- $\hat{U}_p,\ \hat{\mathbf{u}}_f$ are approximation of the original singular vectors (e.g., by using the FFT rSVD)
- Score $\hat{S}_i \rightarrow f(\hat{U}_p,\ \hat{\mathbf{u}}_f)$ is the approximation of Score $S_i \rightarrow f(U_p, \mathbf{u}_f)$
- All norms $||\bullet||$ in the following equation will be 2-norms

We start at the absolute scoring error $|E_S|$ that compares the two SST scores:

$$\begin{aligned}|E_S| :=&|S_i - \hat{S}_i| = |(1 - ||U_p^T \mathbf{u}_f||^2) - (1 - ||\hat{U}_p^T \hat{\mathbf{u}}_f||^2)| \\ =&|(||U_p^T \mathbf{u}_f||^2 - ||\hat{U}_p^T \hat{\mathbf{u}}_f||^2)| \end{aligned} \tag{14}$$

We will adapt (14) until we can use Wedin's general sin-theorem [32] from matrix perturbation theory. The theorem bounds the largest canonical angle between subspaces of the original and perturbed matrices. In our case, we work with the column subspace of H_p/H_f. The left singular vectors $U_p/\mathbf{u}_f$ are an orthonormal basis for this subspace.
As a first step, we want to replace the orthonormal bases U_p^T and $\mathbf{u}_f$ with their corresponding subspace projectors $P_p = U_p U_p^T$ and $P_f = \mathbf{u}_f \mathbf{u}_f^T \in \mathbb{R}^{NxN}$. Before doing so, we prove that this replacement does not change both norms in (14):

$$\begin{aligned}||P_p P_f||^2 &:= ||\underbrace{U_p U_p^T \mathbf{u}_f}_{=:\mathbf{a}}\ \underbrace{\mathbf{u}_f^T}_{=:\mathbf{b}^T}||^2 = ||\mathbf{a}\mathbf{b}^T||^2 \\ &= \lambda_{\max}\left((\mathbf{a}\mathbf{b}^T)^T(\mathbf{a}\mathbf{b}^T)\right) = \lambda_{\max}(\mathbf{b}\underbrace{\mathbf{a}^T\mathbf{a}}_{\text{scalar}}\mathbf{b}^T) \\ &= \mathbf{a}^T\mathbf{a} \cdot \underbrace{\lambda_{\max}(\mathbf{b}\mathbf{b}^T)}_{\star} = ||\mathbf{a}||^2 \cdot \underbrace{||\mathbf{b}||^2}_{||\mathbf{u}_f||^2=1} = ||\mathbf{a}||^2 \\ &= ||P_p \mathbf{u}_f||^2 = ||U_p U_p^T \mathbf{u}_f||^2 = \mathbf{u}_f^T (U_p U_p^T)^T U_p U_p^T \mathbf{u}_f \\ &= \mathbf{u}_f^T U_p \underbrace{U_p^T U_p}_{=I} U_p^T \mathbf{u}_f = \mathbf{u}_f^T U_p U_p^T \mathbf{u}_f \\ &= ||U_p^T \mathbf{u}_f||^2 \text{ q.e.d.} \end{aligned} \tag{15}$$

$\star$: $\mathbf{b}\mathbf{b}^T$ is rank one and has eigenvalue $||\mathbf{b}||^2$

For understanding the proof, note that $||A||^2$ is equal to the largest eigenvalue $\lambda_{\max}$ of $(A^T A)$ [31, p. 69] and $\lambda(cA) = c\lambda(A)$ if c is scalar. Consequently, we can replace U and $\mathbf{u}$ with their corresponding projectors in (8), without changing the result:

$$|E_S| = |(||P_p P_f||^2 - ||\hat{P}_p \hat{P}_f||^2)| \tag{16}$$

We want to apply the reverse triangle inequality $(|(|a| - |b|)| \leq |a - b|)$, but have to deal with the squares $||\cdot||^2$ first. For that we replace $a := ||P_p P_f||$ (positive, scalar) and $b := ||\hat{P}_p \hat{P}_f||$ (positive, scalar).

$$|E_S| = |a^2 - b^2| = |(a-b)\underbrace{(a+b)}_{\geq 0}| = |a-b|(a+b) \tag{17}$$

We can then use the sub-multiplicativity of the Hölder norms $(||AB|| \leq ||A|| \cdot ||B||)$ [31, p. 69] (here $p = 2$) and the fact that all orthogonal projectors have norm of one $||\hat{P}_p|| = ||P_p|| = ||P_f|| = ||\hat{P}_f|| = 1$, since the matrices $\hat{P}_p$ and P_f are idempotent (largest eigenvalue is one) [31, p. 9]:

$$\begin{aligned} a &= ||P_p P_f|| \leq ||P_p|| \cdot ||P_f|| = 1 \\ b &= ||\hat{P}_p \hat{P}_f|| \leq ||\hat{P}_p|| \cdot ||\hat{P}_f|| = 1 \\ &\text{since } 0 \leq a, b \leq 1 \rightarrow (a+b) \leq 2 \end{aligned} \tag{18}$$

Using (18) in (17) and replacing a and b again now yields:

$$|E_S| \leq 2|(||P_p P_f|| - ||\hat{P}_p \hat{P}_f||)| \tag{19}$$

Apply the reverse triangle inequality $(|(|a| - |b|)| \leq |a - b|)$ to get both terms in the same norm:

$$|E_S| \leq 2||P_p P_f - \hat{P}_p \hat{P}_f|| \tag{20}$$

Now add and subtract $\hat{P}_p P_f$:

$$\begin{aligned}|E_S| &\leq 2||P_p P_f - \hat{P}_p P_f + \hat{P}_p P_f - \hat{P}_p \hat{P}_f|| \\ &= 2||(P_p - \hat{P}_p)P_f + \hat{P}_p(P_f - \hat{P}_f)||\end{aligned} \tag{21}$$

Apply the triangle inequality $(||A + B|| \leq ||A|| + ||B||$, for arbitrary matrices $A, B)$ to separate both terms again:

$$|E_S| \leq 2||(P_p - \hat{P}_p)P_f|| + 2||\hat{P}_p(P_f - \hat{P}_f)|| \tag{22}$$

Again, use the sub-multiplicativity of the norm to eliminate the factor in each of the two norms:

$$|E_S| \leq 2||P_p - \hat{P}_p|| \cdot ||P_f|| + 2||\hat{P}_p|| \cdot ||P_f - \hat{P}_f|| \tag{23}$$

with $||\hat{P}_p|| = ||P_f|| = 1$ as explained before:

$$|E_S| \leq 2\underbrace{||P_p - \hat{P}_p||}_{T1} + 2\underbrace{||P_f - \hat{P}_f||}_{T2} \tag{24}$$

Both parts $T1$ and $T2$ are norms of the differences between subspace projectors. Stewart and Sun state that the normed difference is equal to the largest canonical angle between the two subspaces [31, pp. 92&93]:

$$||P - \hat{P}|| = sin\theta_1 = ||sin\Theta|| \tag{25}$$

With this form, we can apply Wedin's Theorem [32, p. 102, (3.1)] which gives us a bound on the canonical angles. In the original equation, one can replace ϵ with the perturbation $||E||$ as noted by Wedin [32, p. 107 below (4.4)]. Since our $||E||$ is random, we use the modified Wedin's theorem of O'Rourke et al. [51]:

$$||P - \hat{P}|| = sin\theta_1 = ||sin\Theta|| \leq 2\frac{||E||}{\Delta} = 2\frac{||E||}{\sigma_k - \sigma_{k+1}} \tag{26}$$

Replacing both $T1$ and $T2$ in (24) with (26), we finally arrive at an upper bound for our scoring error that only depends on the perturbation $||E||$ and the singular values of the original Hankel matrices H_p and H_f (III-I, see (9)):

$$|E_S| \le \underbrace{4\frac{||E_p||}{\sigma_{p,k}-\sigma_{p,k+1}}}_{2\cdot T1} + \underbrace{4\frac{||E_f||}{\sigma_{f,1}-\sigma_{f,2}}}_{2\cdot T2}$$

Note that the factor 2 could be eliminated, if we would use $||U_p^T \mathbf{u}_f||$ without the square to calculate the score.
The rSVD subspace is not literally the top-k subspace of some $A+E$ matrix, but the projection residual $(I-QQ^T)A$ acts as an effective perturbation; treating it this way, as in Saibaba [52], justifies applying Wedin's theorem to bound the canonical angles. This also extends to other subspace decompositions.

APPENDIX. C

We now derive (12) in Section III-I. The result of the previous section is independent of the decomposition method, but we are using the rSVD with a Gaussian random projector. Consequently, we can also employ bounds specialized for randomized decompositions. Saibaba [52] proposes a bound for $sin\theta_i$. Some of their results are dedicated to comparing truncated subspaces from the exact SVD and the rSVD ($sin\theta_j'$ in the paper):

$$sin\theta_1 \le \gamma_1 \frac{\gamma_k^{2q}}{1-\gamma_k}||\Omega_2\Omega_1^\dagger||^2 \tag{27}$$

$$\text{with } \gamma_i := \frac{\sigma_{k+1}}{\sigma_i} \tag{28}$$

Using the inequality $||\Omega_2\Omega_1^\dagger||^2 \le C_e$ in the proof of theorem 6 for Gaussian random projectors [52, p. 39], we arrive at the following:

$$sin\theta_1 \le \gamma_1 \frac{\gamma_k^{2q}}{1-\gamma_k}\left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p}\right) \tag{29}$$

Replace the singular value gaps γ_i:

$$sin\theta_1 \le \frac{\sigma_{k+1}}{\sigma_1}\frac{\left(\frac{\sigma_{k+1}}{\sigma_k}\right)^{2q}}{1-\frac{\sigma_{k+1}}{\sigma_k}}\left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p}\right) \tag{30}$$

By replacing $T1$ and $T2$ in (24) with the previous equation, we arrive at the final bound for the scoring error (III-I, see (12)):

$$\begin{aligned}|E_S| \le & 2\frac{\sigma_{p,k+1}}{\sigma_{p,1}}\frac{\left(\frac{\sigma_{p,k+1}}{\sigma_{p,k}}\right)^{2q}}{1-\frac{\sigma_{p,k+1}}{\sigma_{p,k}}}\left(\sqrt{\frac{k}{p-1}} + \frac{e\sqrt{(k+p)(N-k)}}{p}\right)\\ &+ 2\frac{\sigma_{f,2}}{\sigma_{f,1}}\frac{\left(\frac{\sigma_{f,2}}{\sigma_{f,1}}\right)^{2q}}{1-\frac{\sigma_{f,2}}{\sigma_{f,1}}}\left(\frac{1}{\sqrt{p-1}} + \frac{e\sqrt{(1+p)(N-1)}}{p}\right)\end{aligned}$$

APPENDIX. D

We promised additional thoughts on real and positive semi-definite Hankel matrices. As discussed in the following, they are not guaranteed on our case, but other applications might benefit from our insights.
Beckermann et al. [53] prove that all real and positive semi-definite Hankel matrices have rapidly decaying singular values. Their result is an impressive argument as to why low-rank approximations often work so well for different engineering problems. They even derive an analytical expression for the exponential decay of the eigenvalue spectrum:

$$\frac{\sigma_{i+2l}}{\sigma_i} \le \underbrace{16\left[\exp\left(\frac{\pi^2}{4\log(8\lfloor N/2\rfloor/\pi)}\right)\right]^{-2l+2}}_{=:f_{dec}^l} \tag{31}$$

This fast decay directly shows that rSVD is especially suitable for positive semi-definite Hankel matrices. The analytical expression even allows to compute an extremely cheap estimate of the approximation error. Halko et al. [22] discuss online error estimation in chapter 4.4 of their paper. Their error estimation comes at the cost of several iterative matrix vector products of the original matrix A with a batch of random vectors. Given positive semi-definite Hankel matrices, the results of Beckermann et al. [53] allow for an even cheaper estimation $\mathbb{E}_k^*$. Combining (6) with (31), we can derive a new upper bounded estimation that comes only at the cost of multiplying two constant factors by the largest eigenvalue σ_0. During computation, we could estimate the largest singular value $\tilde{\sigma}_0$ for an approximation of the bound:

$$\tilde{\mathbb{E}}_{\text{Proj}}^* \lesssim \underbrace{f_{proj}^k}_{\rightarrow(4)} * \underbrace{f_{dec}^{k/2}}_{\rightarrow(31)} *\tilde{\sigma}_0 \tag{32}$$

This bound can be used to assess the decomposition quality during runtime. This allows the creation of adaptive algorithms that change their parameters depending on the data properties, allowing to specify an expected approximation error instead of the rSVD parameters. It is especially useful because the largest singular vector can be obtained independently, efficiently and accurately.
Unfortunately, not all possible Hankel matrices from time series are positive semi-definite. To show that a Hankel matrix is positive semi-definite, it is sufficient to show that all eigenvalues are positive (the matrix is real, symmetric; therefore, Hermitian). An example of a Hankel matrix with negative eigenvalues can be constructed from a perfect rectangular impulse of amplitude one in an otherwise zero signal:

$$M_{\text{neg}} = \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 0\end{bmatrix} \quad \text{Eigenvalues: 1, 1, -1}$$

The example shows that positive semi-definiteness does not generally hold for Hankel matrices. Therefore, focusing on SST-based Hankel matrices in the paper, we study the spectral decay of Hankel matrices empirically.

•••

Original Manuscript ID: Access-2025-38816

Original Article Title: “Accelerating Singular Spectrum Transformation for Scalable Change Point Detection”

To: IEEE Access Editor

Re: Response to reviewers

Dear Editor,

Thank you for allowing us to resubmit our manuscript and for the opportunity to address the reviewers’ comments. We are thankful for the reviewers' effort and helpful comments, which we used to improve our submission.

We are uploading (a) our point-by-point response to the comments (below) (response to reviewers, under “Author’s Response Files*”*), (b) an updated manuscript with yellow highlighting indicating changes (as “Highlighted PDF*”*), and (c) a clean updated manuscript without highlights (“Main Manuscript”*).*

Note that addressing each reviewer’s concerns starts on a new page for better readability. The section and figure numbers in this response to the reviewers refer to the original, reviewed version of the paper. In our updated manuscript, which extends the paper by two figures, the figure numbers have changed.

Also, we did some minor text edits (such as line breaks and minor wording) not mentioned in the author actions. These were necessary to avoid unfavorable page breaks after we changed the manuscript to address the reviewers' comments.

Best regards,

Lucas Weber et al.

Reviewer #1, Concern #1:

Original concern: All matrices and vectors are defined on the real field. Why use Hermitian transpose in Algorithm 3.

Author response: The observation is correct; the Hermitian transpose is unnecessary in our context, as all matrices only contain real values. Initially, we kept it for comparability with the source publication [1]. Many of the algorithms (e.g., the fast Hankel matrix product and the rSVD) also work on complex-valued matrices. We, therefore, kept the original Hermitian transpose, which simplifies to a plain transpose for real-valued matrices, to maintain this generalizability.
Nevertheless, we agree that the Hermitian transpose is unnecessary and potentially confusing.

Author action: We thank the reviewer for their attention to detail and changed the Hermitian transpose to a simple transpose. The exact positions of the changes were: equation 1, algorithm 3, the footnote on page 5 of the paper, equation 4, and the inline formulas in Section III-I and at the end of Appendix A.

Sources:

[1] Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." SIAM review 53.2 (2011): 217-288.

Reviewer #1, Concern #2:

Original concern: In Algorithm 4: the matrix C_i is a matrix of size N\times (2N-1). How to ensure the size of matrix C is N\times G?

Author response: In algorithm 4, the matrix \tilde{C_i) is of size (2N-1)x1 (step 6 of algorithm 4). We then extract all elements between (N-1) and (2N-1), resulting in a matrix C_i of size Nx1 (step 7 of algorithm 4). This matrix is a column of the complete matrix C of size (NxG). Step 7 involves two key components: the reverse transformation and the extraction of elements. This is unnecessarily dense, and we will update the algorithm.

Author action: We have improved the textual explanation of Algorithm 4 in Section III-E and separated Step 7 into two steps (see Algorithm 4 in the new version). We also updated the notation in Algorithm 4 to reduce notational overload for matrix C and to maintain consistency with A for arbitrary matrices. This also partially addresses concern #2 of reviewer #3 and concern #2 of reviewer #4.

Reviewer #1, Concern #3:

Original concern: Symbol confusion: in Appendix A, lowercase letters are used to stand for vectors and scalars.

Author response: We agree that it is not straightforward for a reader to distinguish scalars from matrices and vectors in the appendix. We hesitate to change the symbols completely, as they align with the referenced textbooks, and a change might hinder comparability. For our placeholder variables, a and b, in Appendix A, the notation is indeed mixed and inconsistent.

Author action: We updated the manuscript and Appendix A to use bold, lowercase letters for vectors as recommended in the IEEE Access resubmission checklist. The update changes the document in many places. Most changes are for uses of the future singular vector u_f and the vectors used in Algorithm 2, as well as in the text describing the fast Hankel matrix product (Section III-E).

Reviewer #1, Concern #4:

Original concern: In Figure 7, why not consider svd with fft?

Author response: Our answer has two parts: logical and algorithmic. Logically, computing the full SVD incurs significant computational overhead, whereas we only need the first few components to compute the SST. Therefore, it is not worth the effort to compute unused decomposition parts, as their exact calculation requires careful tuning and computationally intensive operations.
Algorithmically, the standard SVD algorithm does not rely solely on multiplications by the original input matrix A, like the rSVD and the IKA-SST, but instead uses a three-step approach [1]. Already in the first step, the matrix is transformed into a bidiagonal matrix, losing the Hankel structure. For this reason, there is no faster version of the complete SVD listed in Figure 7.

Author action: We thank the reviewer for this excellent question, and now explicitly mention the reasoning from our response in Section III-F, where we introduce the FFT rSVD SST (second to last paragraph).

Sources:

[1] Dongarra, Jack, et al. "The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale." SIAM review 60.4 (2018): 808-865.

Reviewer #1, Concern #5:

Original concern: In Algorithm 5: how to choose k for each data set?

Author response: The parameter k is a hyperparameter from the original SST algorithm (see algorithm 1). A potential user of the SST has to choose this hyperparameter independently of our contribution. Our bound (Section III-H/I) shows that k also affects the approximation error we incur when using a more efficient but approximate decomposition. Unfortunately, this still does not allow us to derive any specific, fixed recommendation for choosing the hyperparameter k.
A larger value of k increases computational effort linearly (Table 2), while it decreases the approximation error of the rSVD due to a smaller corresponding singular value (equation 11). We discuss k together with the oversampling parameter p, as p is an additional hyperparameter from the rSVD, which is part of our contribution. The authors of the first SSA-based CPD algorithms [1] and the authors of the SST [2,3] (a further development of [1]) recommend analyzing the singular value spectrum to select a suitable k that captures the signal's most important structure. They recommend setting k large enough to capture the most important structure and small enough to cover only essential elements. In their papers, they mostly mention ks in single digits. [1,2,3]

Author action: We updated Section IV-E (scoring error) to provide better intuition for how we selected k based on our measured singular value spectrum. Still, k remains a hyperparameter of the original algorithm. We also note that in Section IV-E.

Sources:

[1] Moskvina, Valentina, and Anatoly Zhigljavsky. "An algorithm based on singular spectrum analysis for change-point detection." Communications in Statistics-Simulation and Computation 32.2 (2003): 319-352.

[2] Idé, Tsuyoshi, and Keisuke Inoue. "Knowledge discovery from heterogeneous dynamic systems using change-point correlations." Proceedings of the 2005 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2005.

[3] Idé, Tsuyoshi, and Koji Tsuda. "Change-point detection using krylov subspace learning." Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2007.

Reviewer #2, Concern #1:

Original concern: the notation in Sections III-H/I could be simplified or summarized in a short table for readability.

Author response: In Section III-H, the notation is taken from the original source of the error bound [1]. We are hesitant to change the notation to keep compatibility with the original source. Nevertheless, a table summarizing the symbols used in the method is a good idea, and we will update the manuscript accordingly.

Author action: Most significant for the current concern is that we have included a notation table in the appendix, which provides an overview of the notation used in the methods section. Although the table may be larger than intended by the reviewer, we decided that a partial notation table could be confusing. The notation table is separated into several sections for better readability and to help a reader to find symbols faster. We did not include the notation table in the main text because it is relatively large and would significantly reduce readability. The symbols are introduced in the text, so the paper is self-contained without the newly added notation table. Furthermore, combined with the updates addressing the other concerns (from all five reviewers), we have already extended the paper in comparison to the original submission, and we did not want to overuse the main paper's space.
Addressing reviewer #3's concern #1 and reviewer #5's concern #5, we also reworked Sections III-H and III-I. Specifically, we updated the notation and added explanatory text to provide further insight into our intentions for both sections.

Sources:

[1] Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." SIAM review 53.2 (2011): 217-288.

Reviewer #2, Concern #2:

Original concern: a clearer visual summary of the runtime gains (perhaps a single consolidated plot) would improve accessibility for non-specialist readers.

Author response: The gains are the main practical improvement that practitioners will take away from the paper; this point should be easily accessible. Originally, we intended this effect for Figure 7. We agree with the concern and thank the reviewer for this clear suggestion to improve accessibility.

Author action: We thank the reviewer for this suggestion, and have added a new figure (Figure 8b) to consolidate the plot (Figure 7, now Figure 8a in the updated version). New Figure 8b presents a heatmap of runtime factors that better communicates the runtime gains of our improvements to non-specialist readers. As a consequence of adding the figure, we have significantly updated Section III-F, which discusses the runtime gains. This update extends to the conclusion, where we can now reference exact results from Figure 8b (figure number in the new submission). In addition to adding the new figure, we have also updated the descriptions of many figures and tables to clarify their meaning more directly.

Reviewer #2, Concern #3:

Original concern: minor stylistic polishing would help readability.

Author response: We agree.

Author action: For stylistic polishing, we updated our style when referencing equations, sections and algorithms in accordance to [1]. We updated all in-text references of numbered sections to capitalized “Section X”. Only when used in brackets or in Figure 2 (new in the updated manuscript) do we retain the short form for brevity. When the word “section” is used without a specific reference, we keep it lowercase.

We now use the ampersand only in brackets. References to specific pseudocode algorithms are now always abbreviated (alg.), to be consistent with the figures and tables. Finally, we updated our references to equations from “eq. x” to “(x)”. and only use “Equation” when referenced at the beginning of a sentence.

For better readability, we updated the manuscript at several points, addressing reviewer #4's concern #2 (better notation, pseudocode, and careful rewrite) and reviewer #5's concern #3 (proofreading for minor grammatical errors). We updated the axis labels of the figures in the evaluation section (Section IV) and updated the reference sorting (reviewer #5, concern #4). We added more paragraphs within subsections for better readability. Additionally, we ensured that the section breaks are as favorable as possible. In general, we polished the grammar and wording at several positions.

Sources:

[1] https://journals.ieeeauthorcenter.ieee.org/your-role-in-article-production/ieee-editorial-style-manual/, visited 27.10.2025

Reviewer #2, Concern #4:

Original concern: pls discuss possible implications or extensions for multidimensional or streaming time-series data in the Discussion section.

Author response: The SST and IKA-SST are restricted to univariate time series. That said, we agree that parts of the algorithms discussed in the paper, particularly the rSVD and the fast Hankel matrix product, may be of interest for multidimensional methods. At the end of Section IV-G (application example), we briefly mention how increased efficiency might have implications for the online application of the algorithm in a streaming scenario.

Author action: We have updated our discussion in Section V-A to address the possible implications of the proposed algorithms for multilevel Hankel matrices that occur in multivariate change point detection. Additionally, we updated our outlook section (Section V-B) to discuss streaming applications of the SST. We kept the comments concise as they are only implications and not the focus of our paper.

Reviewer #3, Concern #1:

Original concern: Provide a brief formula or illustration that summarizes the theoretical error bound.

Author response: The error bound is presented in its shortest form in equations 9 and 10 of Section III-I (where eq. 10 is the bound combined with the rSVD bound). We already did several steps to shorten the equations, e.g., replacing the sum of singular values with the largest value times the matrix size (Section III-H for the rSVD error bound), and reducing the terms for q ~ log N as recommended [1]. Nevertheless, we agree that the motivation and meaning of the bounds and sections could be explained more clearly.

Author action: We significantly updated Sections III-H and III-I. We added an intuitive motivation for the error discussion, deriving the bound and a textual illustration of its relevance at the end of the section. Additionally, we updated the abstract and corresponding contribution in Section I to better introduce the bound. Similarly, we updated the introduction to the methods Section III to clarify our intentions. This concern is related to concerns #1 and #5 from reviewers #2 and #5, respectively.

Sources:

[1] Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." SIAM review 53.2 (2011): 217-288.

Reviewer #3, Concern #2:

Original concern: Describe the computational method for Hankel products that are produced quickly.

Author response: We agree that the description of Algorithm 4 is not extensive and that a more detailed description would improve the paper.

Author action: We have significantly expanded the textual description of Algorithm 4 in Section III-E. As per our response to Reviewer #1, Concern #2, we additionally updated Algorithm 4 so that line 7 is less dense.

Reviewer #3, Concern #3:

Original concern: Include a numerical overview of approximation mistakes for each dataset.

Author response: The numerical approximation errors for each dataset are shown in Figure 5 of the paper. The figure contains plots for both datasets (synthetic and real), and the results are consolidated in Table 4, also addressing both datasets. Only Figure 6 is restricted to a single dataset. We assume concern #3 addresses this plot.

Figure 6 should be viewed as a side note highlighting the differences between the IKA-SST and the SST. It is a by-product of our measurements. The original IKA-SST authors qualitatively observe, on a single example signal, that IKA-SST underestimates the scores. We happen to have measurements that confirm their statement on a significantly larger time series dataset. In other words, our measurements allow us to visualize this effect on thousands of signals. It is crucial to understand that the observed behavior is not due to our efficiency modifications. We mention the last point in the text explicitly (Section IV-E).
We added the plot because a practitioner might be interested in whether to use the fastest algorithm (our FFT IKA-SST) or the more accurate one that is still fast (our FFT SST), which is why we wanted to use our data to visualize this difference.

As it is a by-product of our measurements but not the focus, we do not want to devote excessive space to a comparison of scoring errors between IKA-SST and SST. We added the plot to support the decision between the two algorithms, independent of our contribution. Our idea was to present this as a brief side note and service to practitioners reading the paper. Therefore, we keep this observation to a single plot. We hope the reviewer understands our concerns regarding the space.

Author action: We updated the text in Section IV-E to clarify that the observation in Figure 6 is a side note that could be useful to practitioners but is not the focus of our evaluation, since the difference between SST and IKA-SST is not part of our contributions.

Sources:

[1] Idé, Tsuyoshi, and Koji Tsuda. "Change-point detection using krylov subspace learning." Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2007.

[2] https://ieeeaccess.ieee.org/authors/submission-guidelines/, visited 24.10.2025

Reviewer #3, Concern #4:

Original concern: For runtime comparisons, provide more specific system configuration information.

Author response: Section IV-B lists the implementation details and system configurations. We agree that the extended system configuration will help with reproducibility, and other papers often also list the OS and Python versions. Nevertheless, we mention at several positions in the paper that our implementation is only experimental with the intention of relative comparisons between the algorithms in our paper, and we recommend using lower-level languages for the fastest possible absolute time.

Author action: We updated the implementation details in Section IV-B accordingly. It now contains more system information (RAM details, OS version, and Python versions).

Reviewer #4, Concern #1:

Original concern: While the technical contribution is clear, the practical relevance remains limited because the study is confined to SST-based methods. The broader change-point detection literature is vast, including scalable non-parametric and online methods designed for high-frequency or multivariate data. To establish utility, the authors should include comparisons with methods including, and not limited to, NEWMA (Keriven et al., 2020), MMDEW (Kalinke et al., 2025), or NPFOCUS (Romano et al., 2023). A short study contrasting computational cost and detection power would make the contribution more meaningful, as well as a discussion about which scenarios SST (and the accelerated SST) is preferable to alternative methods.

Author response: Change point detectors are not universal. As Truong et al. [1] note in their (selective) review: "Cost functions are related to the type of change to detect". While the literature across different fields commonly uses the term change point, the understanding of what it means exactly varies widely. Consequently, comparing several methods with respect to their detection power requires a common definition of change points. Our related work focuses on methods that have a similar definition and method of detecting changes, and extends to papers that propose efficient Hankel matrix decompositions. To our knowledge, few works address the computational bottleneck of SSA-based CPD, we discuss them in our related work section.

The papers mentioned by the reviewer address notions of a change point and data formats that differ from those of SST-based CPD. The referenced methods address multivariate data and changes in distribution. Also, they do not address SST-based CPD in their own related work, even though it existed before their publication dates (since 2005 for the SST and 2007 for the IKA-SST, respectively). Furthermore, the mentioned papers restrict their related work and comparison to closely related algorithms, as we do.
NPFOCUS compares to its predecessor, FOCuS, and includes related methods such as NEWMA, NUNC, Ross, and Wei-CUSUM.
MMDEW compares to AdwinK, D3, IBDD, NEWMA, Scan-B, and WATCH.
NEWMA compares to its predecessor, EWMA, and another method (Scan-B) in the evaluation.
In summary, they also do not cover all available methods. For example, they do not cover RuLSIF [2], which is a common, well cited method for (univariate) distributional CPD.
Importantly, this is not a critique of any of the papers' work, any related work, or their evaluation section. Instead, we mention this to demonstrate how it is common to focus on closely related algorithms, not covering all possible CPD methods. In our paper, we focus on methods that use SSA to extract and compare time series characteristics, whereas NEWMA, MMDEW, or NPFOCUS cover comparisons of distributions.

We agree that collecting and comparing many different change point-detection methods would certainly be interesting for the field of change point detection. A dedicated analysis for practitioners, when to use which algorithm, also addressing the computational cost, would advance the field in general. Nevertheless, this is beyond the scope of our paper and requires a dedicated publication in the form of a review or benchmarking paper. Adding comparisons with methods having other backgrounds and notions of changes, would cause our paper to lose its focus.

If we were to publish a novel method, we would have to delineate its applications and compare it to other competitors in the field. Importantly, we do not propose a new change point detection method or an extension of SST or IKA-SST. Instead, our paper focuses on addressing the computational bottleneck of an established [3] CPD method. We do not change the algorithmic outline, or the scoring function. The method existed before our contribution, and its applicability to specific use cases or applications remains unchanged. We address this point in our discussion at the end of Section V-A.

The reviewer also notes "The practical utility is, however, unknown" in the answer to question 1: Does the paper contribute to the body of knowledge? The SST and IKA-SST are acknowledged methods [3], and our paper improves their computational efficiency. For example, the computational complexity of the SST is specifically addressed as a problem in [4] and in the paper that published the IKA-SST [5]. While we cannot demonstrate all potential applications within the scope of this paper due to space constraints, we illustrate the practical utility with an application example in Section IV-G. Of course, this does not cover all potential use cases, but we hope this illustrative evidence demonstrates how our contribution has practical utility. We also believe that our improvements not only benefit existing use cases but also enable the SST to be used in scenarios where the window size was previously prohibitively large.

In conclusion, our paper focuses on addressing computational bottlenecks of the SST algorithm. We do not contribute a novel method or alter the scoring, but rather improve the efficiency of an established detection method. Our paper is dedicated to justifying our algorithmic choices, both theoretically and empirically. Our evaluation demonstrates practical utility by significantly reducing algorithmic runtime, while only incurring minor approximation errors. It does not alter the strengths or weaknesses of the original (IKA-)SST. Instead, it speeds up existing use cases and possibly extends the applicability to novel use cases, where it was prohibitively slow before.
Our related work is focused on our contribution similar to the papers mentioned in the review. We agree that comparing and appropriately benchmarking several methods would be of great interest to the community. However, we believe this requires a dedicated paper.

Author action: We have updated the manuscript to clarify that we do not intend to propose a new algorithm or alter the use case of the SST, but rather to enhance the efficiency of an existing algorithm without modifying its strengths or weaknesses in terms of detection capabilities. Specifically, we updated our goal in the introduction (Section I), and the related work section (Section II & II-A). Additionally, we now refer the reader to the sources that discuss the applicability of SSA-based CPD methods at the end of our discussion section (Section V-A).

Sources:

[1] Truong, Charles, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods." Signal Processing 167 (2020): 107299.

[2] Liu, Song, et al. "Change-point detection in time-series data by relative density-ratio estimation." Neural Networks 43 (2013): 72-83.

[3] Aminikhanghahi, Samaneh, and Diane J. Cook. "A survey of methods for time series change point detection." Knowledge and information systems 51.2 (2017): 339-367.

[4] Zhang, Shenglin, et al. "Rapid and robust impact assessment of software changes in large internet-based services." 11th ACM Conference on Emerging Networking Experiments and Technologies. 2015.

[5] Idé, Tsuyoshi, and Koji Tsuda. "Change-point detection using krylov subspace learning." Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2007.

Reviewer #4, Concern #2:

Original concern: The paper’s structure and notation could be clearer. The introduction moves quickly across methodologies, the pseudocode and formulas use inconsistent symbols, and some algorithms are hard to connect. Important details or where algorithms interact are not always explicit. A careful rewrite to unify notation, explain algorithm flow, and document all adaptations would greatly improve readability. For example: N is the matrix side on page 1, but it's nowhere mentioned before page 4 that this corresponds to the window size, or it's not clear where algorithms 2 and 3 come into play until algorithm 4. In algorithm 5, you call a "slightly adapted reference implementation" of algorithm 4, but I could not seem to find this? These are few representative, not exhaustive cases of presentation and reproducibility issues.

Author response: We thank the reviewer for this careful dissection of structural concerns and imprecise notation. We agree that explaining the algorithmic flow is important and will update our paper accordingly. Regarding the specific examples, N is introduced as the window size in the abstract and in the second column on page one. Nevertheless, it is not immediately clear that the window size is a main hyperparameter of the

SST and that it directly influences the matrix size. Also, we agree that the reference in algorithm 5 to a “slightly adapted version” of algorithm 4 has to be clarified.

Author action: For an improved explanation of algorithmic flow, we updated the beginning of our methods (Section III) to explain the algorithmic flow and how the algorithms are connected before introducing them. Most importantly, we now provide a figure that visualizes the interactions of the different algorithms in the methods section. We improved the explanation at the beginning of Section III-F (improving the SST), on how the algorithms interact.

For unifying notation, we updated our notation throughout all sections. Specifically, we revised the notation in Section III-E to avoid overloading matrix C and to maintain the use of A for arbitrary matrices. These updates include algorithm 4 (also see reviewer #1, concern #2, and reviewer #3, concern #2). We reworked sections III-H and III-I (also addressing reviewer #2's concern #1), and changed the Hermitian transpose to a normal transpose (reviewer #1, concern #1). Our appendix now contains a table that provides an overview of the notation (see reviewer #2, concern #1). We updated the notation in algorithm 5 in accordance with the other updates, and we corrected the imprecise reference to the “slightly adapted algorithm 4”. Finally, we updated vector notation to bold symbols as required by the resubmission checklist (also, see concern #3, reviewer #1).

Regarding the introduction of N, we updated the introduction (Section I) to better introduce the window size and its connection to the matrix size for the SVD.

Unrelated to the notational updates, we corrected algorithm 2, where we found an issue with the presentation compared to the original paper. The r in the original paper [1] corresponds to our k, and k in the original paper is used as the Lanczos rank. We now introduce zeta for the Lanczos rank in our paper to maintain consistent notation with the k used in algorithm 1. Our implementation and measurements reflect this difference, but our pseudocode did not.

Sources:

[1] Idé, Tsuyoshi, and Koji Tsuda. "Change-point detection using krylov subspace learning." Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2007.

Reviewer #5, Concern #1:

Original concern: Page 1, Keywords: "Change detection algorithms" is listed, but the text often uses "Change Point Detection." Consider consistency.

Author response: The keywords on page 1 have to be selected from a predefined list upon submission to the IEEE Access website. Change point detection is not a valid option in this list. A large review of time series change point detection [1] also uses both terms (‘change detection’ and ‘change point detection’), and there is no clear definition of how they differ or whether they can be used interchangeably.
Our interpretation of this review is that change detection is often based on change point detection algorithms. This is why we selected change detection as the closest option to change point detection on the submission website. In the paper itself, we consistently use 'change point detection,' not 'change detection.' Nevertheless, we are open to adjusting the selected keywords on the submission website.

Author action: We deselected Change Detection from the keywords during submission.

Sources:

[1] Aminikhanghahi, Samaneh, and Diane J. Cook. "A survey of methods for time series change point detection." Knowledge and information systems 51.2 (2017): 339-367.

Reviewer #5, Concern #2:

Original concern: Page 5, Contributions: "Log-Linear SST & IKA-SST with linear memory." This is a bit informal for a contribution headline. Consider: "Log-Linear Complexity with Linear Memory Requirements."

Author response: We wanted to keep the main contribution point concise, but we recognize its informal tone. Thank you for providing a recommendation.

Author action: We updated this contribution item text to make it more formal.

Reviewer #5, Concern #3:

Original concern: Please perform a thorough proofreading for similar minor grammatical errors and hyphenation (e.g., "change point" vs. "change-point").

Author response: We will review the text accordingly. To potentially explain our mixed notation, examples like “change-point detection” and “change point detection”, or “time-series analysis” and “time series analysis” are also often commonly mixed in the related work. As a consequence, we might have mixed the expressions. Of course, this is no excuse, and we aim to be consistent. Your comment is correct.

The IEEE style guide referenced in [1] requests hyphenation for compound nouns that contain a noun and an adjective, such as sliding-window FFT. For compound nouns that consist of two or more nouns, they only recommend hyphenation in cases where there are many words or it is absolutely necessary for clarity. It also recommends checking the Merriam-Webster dictionary, but at least the online version does not contain words like change point detection or time series analysis.
Therefore, we hyphenate mainly noun-adjective compound nouns and leave others without a hyphen. In accordance with the style guide, we do not hyphenate words like fast matrix product, as the adjective fast does not refer to “matrix” but to “product”.

Author action: We updated the manuscript to ensure consistent wording, hyphenation, and correction of minor grammatical errors. We do not list all instances of changes in this answer, as the changes cover almost all sections. Please see the updated manuscript (and the version with marked changes).

Sources:

[1] https://journals.ieeeauthorcenter.ieee.org/your-role-in-article-production/ieee-editorial-style-manual/, visited 27.10.2025

Reviewer #5, Concern #4:

Original concern: Please correct and sort reference numbers in the text.

Author response: We thank the reviewer for their attention to detail. Section IV-E contains an incorrect, literal citation, where we unfortunately cited an older paper of the same authors. Regarding the ordering of reference numbers, there was an error in our bibliography style configuration (BibLaTex did not properly use the IEEEtran.bst).
We are unsure whether this concern also refers to the order of our appendices. Appendix C is referenced before Appendices A and B. We propose to keep this order, as Appendix C discusses positive semi-definite Hankel matrices, which are only a side note and the content is the least relevant to our paper.

Author action: We corrected the literal citation and ordered the remaining citations by appearance. There was an error in our bibliography style configuration, which we have now corrected. Primarily, this affected the citation numbers in the related work, which are now listed in the order of their citation.

Reviewer #5, Concern #5:

Original concern: Improve the exposition in Section III-I (Connecting Decomposition and Scoring Error) to provide more intuition for the theoretical bounds before presenting the equations.

Author response: We agree that including additional intuition helps with understandability.

Author action: We significantly updated Section III-I. The section now includes an intuitive motivation before listing the formulas. At the end of the section, we now illustrate the relevance of the derived bounds. Additionally, we updated Section III-H (decomposition error) to motivate our subsequent discussion and specifically address Hankel matrices as they occur in the SST. These changes provide a better intuition and a smoother transition into Section III-I.
We also modified mentions of the bound in the abstract, in the contributions (Section I), and at the beginning of the methods Section III, to better guide the reader. This is related to reviewer #2's concern #1 and reviewer #3's concern #1.

Note: *References suggested by reviewers should only be added if it is relevant to the article and makes it more complete. Excessive cases of recommending non-relevant articles should be reported to ieeeaccesseic@ieee.org*